Placement and Routing for Cross-Referencing Digital Microfluidic Biochips

XIAO, Zigang

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong

October 2010


香港中文大學
THE CHINESE UNIVERSITY OF HONG KONG
23 APR 2013
LIBRARY SYSTEM

Committee Members

Prof. YOUNG Fung Yu (Supervisor)
Prof. WU Yu Liang David (Committee Chairman)
Prof. LAU Lap Chi
Prof. CHEN Hung Ming (External Examiner)

Abstract

Digital Microfluidic Biochip (DMFB) has drawn lots of attention today. Through manipulating discrete 'droplets' that containing biochemical materials on the biochip, various kinds of operations can be performed. It offers a promising platform for miniaturized biochemical experiments. Hence, it is also known as 'lab-on-a-chip'. As an increasing amount of highly complicated and concurrent procedures are modeled and performed on biochips, the overall complexity of biochips are expected to increase significantly. Early biochip devices adopted the direct-addressing electrode manipulation scheme, in which each cell on the biochip can be addressed independently. For large arrays, this schemes inevitably introduce a large number of control pins, that can lead to a high product cost. To address the above issue, several pin-constrained biochip designs are proposed. Among them, cross-referencing driving scheme is an elegant design that has a reasonable balance between flexibility and production cost. In this scheme, the electrodes are addressed in a row-column manner. The control of an $N \times M$ grid array can then be achieved by using only $N + M$ control pins. This scheme not only brings down manufacturing cost, but also allows large-scale chip design. Nevertheless, extra cells may be activated during simultaneous movements of multiple droplets under this scheme, which may in turn cause electrode interference and affect droplet routing. Consequently, the parallelism may be severely decreased, which leads to a low throughput. Thus, effective design automation technique is in urgent need to ensure the correctness of droplet movements while maintaining a high throughput.

Most of the previous works on computer-aided design problems for biochips assume the availability of direct-addressing scheme. In this thesis, we formulate and study the droplet routing problem and the placement problem on cross-referencing biochip, and propose automation algorithms to solve them.

We first present a method that solves the droplet routing problem directly.

An initial ordering of the nets will first be obtained according to their relationships. A weighted maze routing algorithm will then be performed to route each net. The stringent electrode interference is modeled as electrode constraint, and is detected and prevented by using an elegant two-coloring graph method during routing. Probabilistic-based rip-up and re-route process is included to break tie and to search for a better routing order. Real-life benchmarks are used to evaluate the proposed methods. Compared with previous work, our router improves on average 4% in routing time and 58% in runtime. It can route all the benchmarks within the time limits, while the latest work fails at some cases.

For the placement problem, we propose an Integer Linear Programming (ILP) based method. Our method considers the characteristics of cross-referencing biochip and is aware of droplet routing. The location of the dispensers and reservoirs will be first computed and fixed. Next, the placement problem will be formulated as an ILP and solved using an ILP solver. Pin assignment is performed at the end. Experimental results show that by running our router on the placement solution generated by our method, an average improvement of 11%, 29%, 54% and 46% in the latest arrival time, average routing time, stalling steps and cell usage respectively can be achieved in comparison with the placement generated by the latest work on this problem.

摘要

數字微流控生物芯片是現時的一個研究熱點。通過在芯片之上控制含有特定化學物質的離散液滴，使其進行各種操作，許多生化實驗都可以在這個平台上進行，因此，它也被稱作"晶片實驗室"。隨著越來越多的複雜生化實驗被成功建模，並在生物芯片上实现，可以預見生物芯片的整體複雜度將會越來越庞大。早期的生物芯片使用了電極直接驅動的設計，芯片上的每一個單元都能被獨立驅動且互不干擾。然而，對於大型的生化實驗，採用這種設計的生物芯片必須使用大量的驅動引腳，從而帶來高昂的生產成本。為了解決這個問題，人們提出了幾種限制引腳的電機運作設計。其中，引腳交叉引用驅動的設計在操作靈活性以及生產成本之間達到了合理的平衡，是一種優秀的設計方案。這種設計使用了一種行-列引腳交叉驅動的方法來操作電極。對於原來要使用$N \times M$ 個控制引腳的陣列，在這種設計中只需要$N + M$的數量。因此，生產成本可以得到有效的減少，從而大規模的芯片設計也更可行。然而，當在這種設計中同时操作多个液滴，可能會多餘的電極被驅動，它們會產生電極干擾並影響到液滴的移動。因此，液滴移動的並行性可能會嚴重降低，從而帶來低生產率。所以，研究有效的設計自動化技術，保證液滴的正確移動並且維持高生產率，勢在必行。

此前，大多數對於生物芯片的計算機輔助設計研究工作都是基於電極直接驅動的設計而進行的。在此篇論文裡，我們將在使用交叉驅動設計的生物

芯片的基礎上，對液滴佈線問題以及布圖問題做一個形式化的表述，並且提出了解決這兩個問題的方法。

對於液滴佈線問題，我們設計了一種能夠直接解決該問題的算法。首先，根據連線之間的關係，可以計算並得到一個初始的佈線排序。然後，依次對每個連線使用一個加權的迷宮尋徑算法而得到佈線。我們的佈線器使用了一個基於圖二染色的算法來檢測並預防上述提到的電極干擾。同時，我們還設計了一個基於概率的取消與重布過程以解決佈線中可能遇到的死鎖問題，並得到更好的佈線排序。我們使用了真實的測試數據來對方法進行評測。對比起以往的方法，我們的佈線器在液滴路由時間以及運行時間上分別有了平均4%和58%的性能提升。同時，我們的佈線器可以在限定時間內完成所有液滴的佈線，而被比較的方法則在某些測試數據下無法得到符合時間限制的解。

對於布圖問題，我們提出了一種基於整數線性規劃的方法。我們的算法考慮了交叉電極設計的特性以及佈線。我們的方法先計算並且固定下儲液器的位置。然後，對於正在解決的布圖問題實例，可以建立起一個整數線性規劃，並且使用求解器進行求解。引腳位置將在最後得到分配。實驗數據表明，在我們的布圖結果上使用我們的佈線器，以及在最新的方法得到的布圖結果上使用我們的佈線器，對於液滴最遲到達時間，液滴路由時間，液滴停止次數以及單元使用數，分別有平均11%, 29%, 54%, 46%的性能提升。

Dedicated to my parents

Acknowledgement

I would like to show my deepest gratitude to my supervisor, Professor Evangeline F. Y. Young, whose guidance and encouragement from the initial to the final level enabled me to develop and complete the thesis. Without her constructive advices and thoughtful comments this thesis would not have been possible. I am heartily thankful to her for receiving her excellent supervision in these two years. Her wide knowledge and her logical way of thinking have been of great value for me. I could not have imagined completing my master's study without her patience, motivation, enthusiasm and continuous support.

My sincere thanks goes to my dear group members Linfu Xiao, Zaichen Qian and Yan Jiang, who shared valuable comments for the thesis. Special thanks to Liang Li, who provided invaluable advice when I was choosing supervisor and research field, and offered lots of helps when we were living as neighbors. I am also grateful to my colleagues in VLSI EDA/Testing laboratory. I enjoyed every single day spent with you.

I am indebted to my father Xianzhi Xiao and Liping Yuan. It is you that inspired me spiritually and offer me the freedom to choose my own way and pursue the goal I dream for.

Lastly, I offer my blessings to all of those who supported me in any respect during the completion of the thesis. My apology for unable to mention each of you personally.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Microfluidic-based Biochip has received more and more attention today [8]. With the help of integrated circuit technology, nano-level biochemical material can be transported and processed in the form of tiny 'droplets'. Precise control of nano-liter droplets of biochemical reagents, samples and buffers are available in these devices. As a result, basic operation unit such as *mixing, detection* and *diluting* can be built. Consequently, various kinds of biochemical reaction can be modeled and performed on this platform. Hence, biochip can be viewed as miniaturized laboratory, i.e., 'lab-on-a-chip'. It shows great advantages in medical, pharmaceutical and environmental monitoring applications [9]. Instances include immunoassays for point-of-care medical diagnostics, DNA sequencing, and detection of airborne particulate matter [9, 10, 11, 12, 13, 14, 15, 16]. In contrast to conventional expensive and tedious laboratory procedures, advantages of miniaturized biochips include higher sensitivity, lower cost due to smaller sample and reagent volumes, higher levels of system integration, less human resource and less likelihood of human error [17]. As a result, markets are opening up for such kind of devices. For example, the worldwide market for *in-vitro* diagnostics in 2007 was estimated at $38 billion [18]. The remarkable market indicates a prospective future of biochips.

The rest of this chapter is organized as follows. The microfluidic technology used in biochips is introduced in Section 1.1. Pin-constrained devices that use a limited number of electrode for manipulation to reduce manufacturing cost are introduced in Section 1.2. Section 1.3 gives an overview of the CAD problems related to biochip. Next, the placement problem and the droplet routing prob-

lem on cross-referencing biochip will be discussed in Section 1.5 and Section 1.4, respectively. Sections 1.6 describes the contributions of this thesis. Finally, the organization of the remaining thesis is presented in Section 1.7.

1.1 Microfluidic Technology

Early implementation of microfluidic biochips relied on the continuous flow microfluidic technology. Commercial biochips of this type are also available in the market. Later, digital microfluidic technology was developed to overcome the disadvantages of continuous flow microfluidic system. In this section, we will give a brief introduction to both systems.

1.1.1 Continuous Flow Microfluidic System

Traditional biochips utilizes continuous flow microfluidic technologies. In continuous flow systems, flow of liquid are piped through micro-fabricated channels. The pumping is performed by either external pressure sources, integrated mechanical micropumps, or electrokinetic mechanisms [19]. Simple and well-defined application are made available in these systems. However, they are not suitable for highly complicate fluid manipulation because of the inherent nature of continuous flow. The parameters that govern flow field (e.g., pressure, fluidic resistance, etc.) vary along the flow-path, which makes the flow dependent on the whole system and thus hard to control. The behavior is even more complicated when the operation is going on, as the electrical and hydrodynamic properties of liquids vary during the mixing and reaction. Hence, the design and analysis of these systems is barely tractable even for a moderate complex application . In addition, the system has to be custom-made according to each specific application, because of the tight coupling of the system implementation and the functionality.

1.1.2 Digital Microfluidic System

A more promising technology is digital microfluidic system. Electrowetting on dielectric (EWOD) are utilized to manipulate and move discrete nanoliter objects that contain chemical materials on a two-dimensional electrode array. These ob-

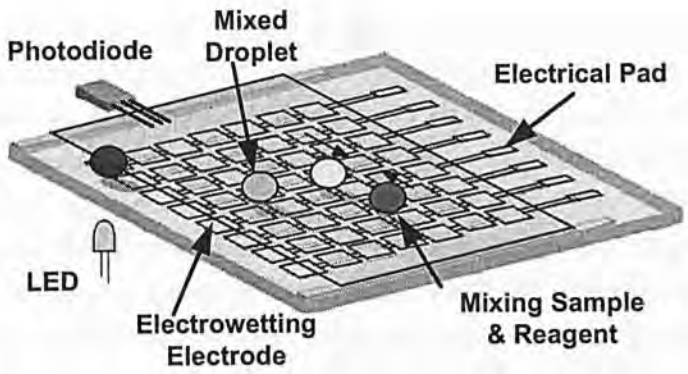


Figure 1.1: Schematic view of DMFB.

jects are called *droplets*, and they are the basic units to manipulate on a biochip. The droplets can be imported onto the chip via *dispensers* ports, and can be exported via *waste reservoir* ports. There is a parallel pair of top and bottom plates of electrodes in the basic design of such systems. The bottom plate contains a patterned array, in which each electrode can be addressed independently, while the top plate is filled with ground electrode. A droplet can be moved to one of its four neighbor cells by applying a control voltage over the target cell, while at the same time deactivating the electrode at the current position of the droplet. An interfacial tension gradients will be generated by the electronic manipulation, and the droplet will be dragged to the activated electrode. Consequently, the droplet can be moved to any place of the array. Furthermore, detectors like photodiodes and LEDs can also be integrated into the biochips to perform optical monitoring tasks. Figure 1.1 gives a schematic diagram of a digital microfluidic biochip (DMFB) [5].

Compared to continuous flow systems, digital microfluidic systems improve in several aspects. First of all, the digitized droplet-based transportation methods allow more versatile manipulation of operations. Complex system can be built due to the decomposition of structure and functionality. Secondly, different basic operations, e.g., mixing, spliting, etc., can be modularized by just applying different sequences of control voltages. For example, mixing can be achieved by moving two droplets together first, and move the combined mixture towards some pivot point repeatedly to complete the desired action. In the old-fashion device, dedicated on-chip reaction sites must be designed in fixed places, while in digital microfluidic biochips, the fluidic operations can be conducted anywhere on the chip. This important distinction is the *reconfigurability* offered by the digital microfluidic

biochips. As a result, general purpose biochips are made available. Different bioassays and applications can be modeled and translated to a corresponding schedule of droplet movements, and programmed into the on-chip microcontrollers that can implement the moves of droplets. Mass production with lower production cost for multi-functional chips are thus possible.

Nevertheless, this emerging technology is still in its infancy. Chip and system integration is in urgent need. Particularly, design methods are needed to ensure that biochips are as versatile as macro-labs that are intended to be replaced [17]. The reconfigurability of digital microfluidic biochips are insufficiently exploited. Most of the commercial biochips available today are designed specifically for particular applications. The users do not have the flexibility to design and implement their own experiments. Hence, computer-aided design techniques are strongly needed to bridge the gap. Growing interest has been seen in automated design and synthesis of biochips from bioassay protocols in recent years.

1.2 Pin-Constrained Biochips

An important design consideration is the electrode manipulation method. It refers to the manner that the electrodes are controlled and activated by input pins. Early designs of biochips use direct-addressing manipulation scheme, in which every electrode is associated with an input pin. In this scheme, an electrode is fixed under each cell of the chip. Each cell can thus be addressed independently to bring about the droplet movements, i.e., pulling a droplet to it from one of its adjacent cells. Although this scheme is straightforward and simple, it is not suitable for large array applications for a few reasons. First of all, the need of a large number of control pins will increase the production cost significantly. Secondly, the interconnect routing problem is also complicated by these control pins. A number of pin-constrained designs are proposed to leverage the drawbacks of this direct-addressing scheme. Pin-constrained addressing scheme maps the electrodes to a smaller number of control pins, i.e., each pin controls more than one electrodes. In the following sections., descriptions of three types of pin-constrained biochips are introduced.

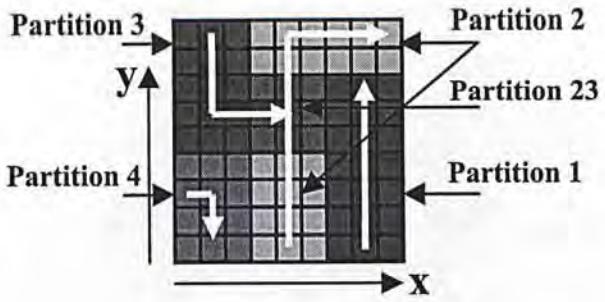


Figure 1.2: Illustration of droplet-trace-based array partitioning method [1].

1.2.1 Droplet-Trace-Based Array Partitioning Method

In paper [1], a droplet-trace-based array partitioning method is proposed. The set of traversed cells of a single droplet is referred to as the 'droplet trace' in the paper. Their method tries to partition a biochip array into regions in such a way that at most one droplet is included in each region. The droplet traces will be the initial partitions. Direct-addressing will be used if two droplet traces intersect with one another for otherwise, electrode interference may be caused. An example is given in Figure 1.2. A 'connect-5' algorithm is proposed in paper [1] to generate the pin assignment. However, this design has at least two disadvantages. Firstly, the partitioning is based on the routing information of the droplets, which means that the pin connection needs to be fixed after the physical synthesis of the biochips. Therefore, the array design is limited to a specific bio-assay. Secondly, the proposed pin-constrained design can only work well under simple bio-assays. In complicated bio-assays, the routings of droplets tend to create a lot of intersections in the congested area. Direct-addressing will then be needed in those intersection cells. As a result, the reduction of pins may be little in this case.

1.2.2 Broadcast-addressing Method

Another pin-constrained technique is proposed in [2]. The authors discovered that *compatible sequences* of electrode activation may exist in any specific bio-assay. Then, the number of control pins can be reduced by connecting compatible electrodes together to a single control pin. In other words, the activation of a pin will activate a set of connected electrodes. No problem will occur if the design is done delicately and correctly. This method is thus named as 'broadcast-addressing'

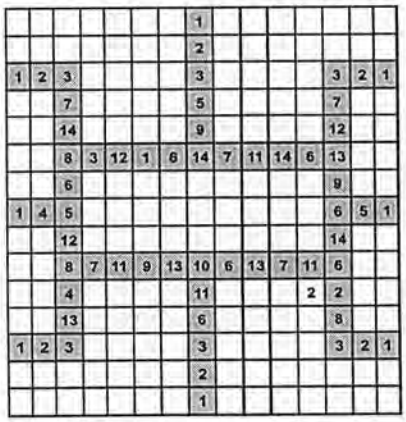


Figure 1.3: Illustration of broadcast addressing for the PCR chip [2].

for the one-to-many mapping. An example is given in Figure 1.3. Similar to the method in Section 1.2.1, this method also requires droplet routing information beforehand. To address this, the author proposed a way that allows a set of predetermined bio-assays to be executed on the same biochip using the broadcast-addressing method. For each bio-assay, the droplet routing information is first collected. Next, a collective activation sequence of an electrode can be obtained by merging the activation sequences from all assays. It is obvious that tradeoff exists between the number of bio-assays included in the biochip and the number of control pins. The more bio-assays are included, the less flexible the mapping of the pins and thus the larger the number of pins needed. In the worst case, the mapping is one to one as in direct-addressing, and no pin can be reduced.

1.2.3 Cross-Referencing Method

A serious drawback of both methods in previous sections is that the reconfigurability of the biochip is sacrificed to reduce the pin number. Another more promising method is called *cross-referencing*, which applies high and low (or low and high) voltages to a row electrode x and a column electrode y respectively so as to *activate* the cell at the intersection point (x, y). Compared to direct-addressing design that needs $N \times M$ control pins, the cross-referencing biochip only needs $N + M$ control pins. In contrast to the aforementioned pin-constrained designs, cross-referencing electrode manipulation method is a general design that are not restricted to a specific bioassay. A droplet can be moved to anywhere on the chip.

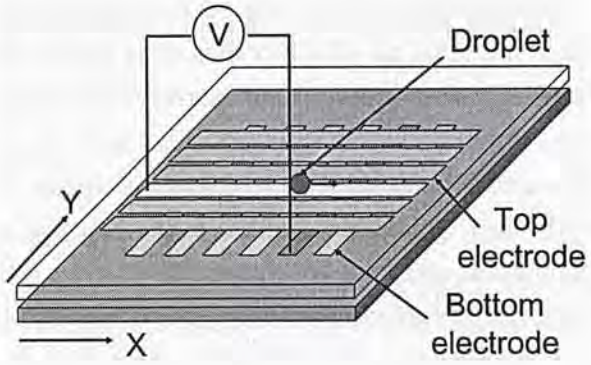


Figure 1.4: A Cross-referencing biochip.

The reconfigurability of digital microfluidic biochip is preserved. Figure 1.4 gives an illustration of a cross-referencing biochip [20]. However, because the activation of a cell is in a row-column manner, *extra cells* may be activated when several droplets are moving together. Thus, the droplets may be affected unintentionally. This unwanted effect is referred as *electrode interference*, and *electrode constraint* should be imposed to avoid such erroneous cases during droplet routing. Cross-referencing imposes tighter constraints to simultaneous manipulation of droplet movements than direct-addressing. Nevertheless, the parallelism can be maximized by carefully arranging voltages. This will be further discussed in the following section.

1.2.3.1 Electrode Interference in Cross-Referencing Biochips

In cross-referencing biochips, *extra cells* may be activated when more than one droplets are going to be manipulated at the same time. These extra activated cells may in turn affect the movements of the droplets. An example is given in Fig. 1.5. Suppose the left droplet needs to be moved from (1,3) to (1,2) and another from (4,3) to (4,4). We assign high voltage to row 2 and 4, as in Fig. 1.5(b), while assigning low voltage to column 1 and 4, so as to activate both cells (1,2) and (4,4). However, cells (1,4) and (4,2) will also be activated, and they are *extra-activated.* Hence, the droplets may not be moved as planned. If appropriate voltage assignment can be performed as in Fig. 1.5(c), correct movements can be guaranteed. Fig. 1.5(d) is an alternative solution by flipping all the electrodes in Fig. 1.5(c) to its opposite value. For rows and columns which are not marked as 'H' or 'L',

ground voltage is assigned and no cells will be activated along those rows and columns. Note that those extra-activated cells do not necessarily imply electrode interference. If no droplet is around the extra-activated cell, no electrode interference will ever happen. The main challenge of routing droplet in cross-referencing DMFB is to avoid electrode interference. In this sense, a cross-referencing DMFB has much tighter constraint than a direct-addressing one, which means that the throughput may be severely decreased. As the example suggests, however, with the help of a well-designed router, cross-referencing biochip can still achieve high-throughput that can be as good as that in a direct-addressing biochip.

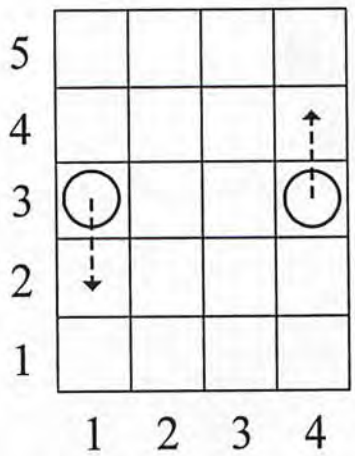
1.3 Computer-Aided Design Techniques for Biochip

As the advancement of microfluidic-based technology leads to more usable biochips available in the market, computer-aided design support is strongly needed for biochips as in traditional VLSI design. Most of the existing works followed the top-down design methodology proposed in paper [21]. In the paper, the design of biochips is divided into *architectural-level* and *geometry-level synthesis*. The behavioral model for a biochemical assay is first acquired from the laboratory protocol and modeled as a sequencing graph. Then, architectural-level synthesis follows to generate the macroscopic structure of the biochip, which contains the *task scheduling* and *resource binding* information. Finally, geometry-level synthesis is performed to generate the detailed layout of the biochip, which includes the module *placement* and *droplet routing*. Analogous to traditional VLSI design, this methodology decouples the complex design flow into several tractable sub-problems. Designers can thus solve the whole problem in several manageable pieces. A flow diagram is given in Figure 1.6 to better illustrate the idea.

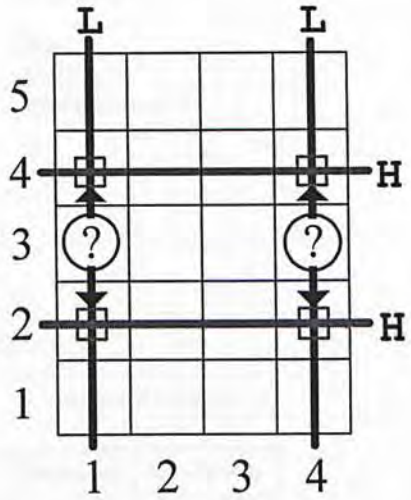
In our thesis, we dedicate to solve the placement problem and droplet routing problem on cross-referencing digital microfluidic biochips. Introductions about these problems will be covered in the following two sections.

1.4 Placement Problem in Biochips

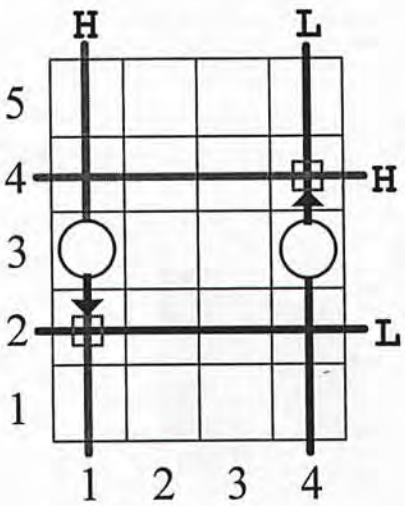
Given a scheduling and a resource binding generated from the architectural-level synthesis, the placement problem involves placing the microfluidic modules in or-



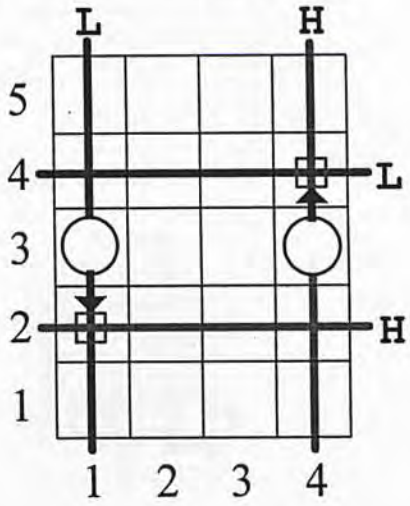
(a) The two droplets are moving from current cell to the arrow-pointed cell.



(b) Electrode interference happens, droplets' movements are intervened.



(c) Solution: apply voltage appropriately.



(d) An alternative solution.

Figure 1.5: Example of electrode interference and solutions.

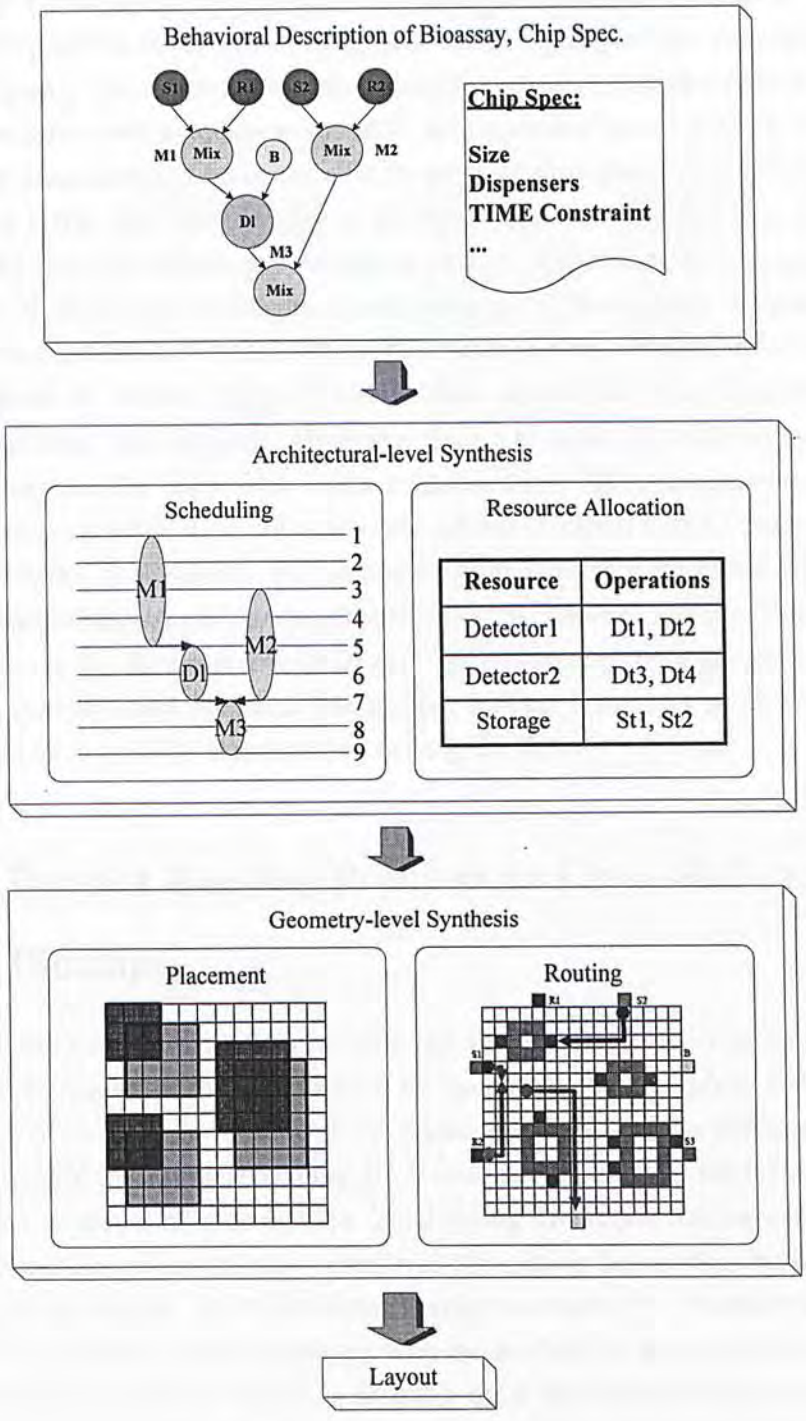
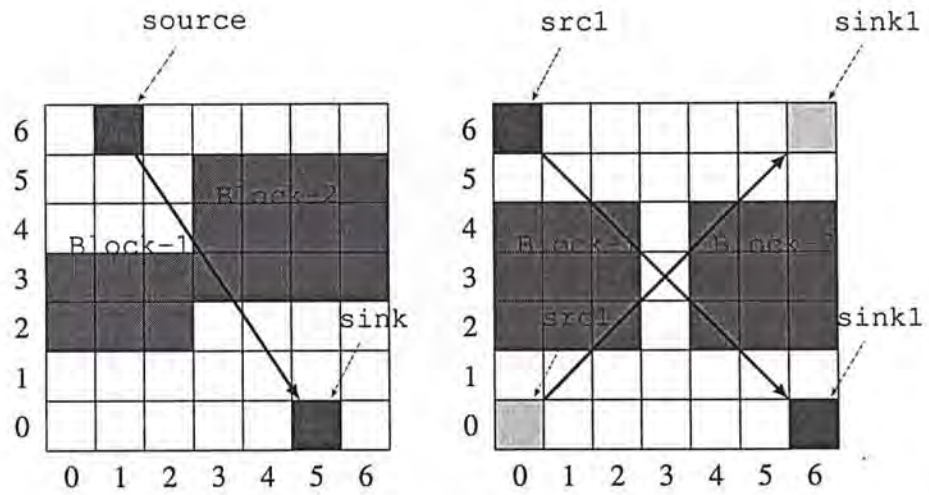


Figure 1.6: Flow of biochip design.

der to optimize some design metrics while satisfying various constraints. It is a critical step before the droplet routing step, which directly affects the routability, routing quality and success of the bio-assay. One of the unwanted solutions is that the placement result is totally unroutable, as shown in Figure 1.7(a). In another case, the placement has a poor quality in terms of throughput. For example, as in Figure 1.7(b), the modules may be placed in an over-congested way that one droplet have to wait for another droplet to pass by. The routing thus becomes sequential. Note that the DMFB placement problem is different from the traditional placement problem in electronic design [22] because of the reconfigurability. First of all, there are various types of modules that can be placed in the same location in different time intervals. Moreover, since a bioassay has a timing span and precedence over the operations, it can be divided into different subproblems according to the scheduling result. Each subproblem is related with its predecessors and successors. In this thesis, we propose a comprehensive method that solves the placement problem by considering the properties of cross-referencing DMFB. The motivation is that the properties observed from cross-referencing is easy to model into the ILP objective function. Meanwhile, feasible result can be obtained and improved in an iterative manner when solving the ILP.

1.5 Droplet Routing Problem in Cross-Referencing Biochips

The droplet routing phase follows after the placement phase. The output of a placement consists of the locations of the operations, i.e., modules, and the pin locations of the droplets. Before an operation can take place, the droplets have to be transported to the corresponding pin locations. Valid routes for them without causing unexpected mixture must be found during the droplet routing step. Meanwhile, a certain amount of time is reserved for routing the droplets between two successive operations, which forms the *timing constraint* [5]. Although this time interval is relatively small comparing with the duration of the operations in each sub-problem (e.g., 0.2s vs 2s), it is desirable to be minimized to prevent spoilage and ensure correctness of subsequent operations. During droplet routing, at least one cell should be kept between droplets to prevent unintended mixing, except



(a) Two modules block the path of the net. This placement is unroutable.

(b) Although the path is not blocked, the two droplet has to coordinate which one should go first. Otherwise, they will form a deadlock in the critical path.

Figure 1.7: Example of poor quality placement.

when droplets are to be mixed intentionally. Hence, *static fluidic constraint* and *dynamic fluidic constraint* are introduced between two droplets [5] to restrict the droplet locations. Furthermore, recall that high and low (or low and high) voltages need to be assigned to the row and column electrodes without causing electrode interference. This can be modeled as *electrode constraint* to ensure the correctness of droplet movements. Finally, the number of cells used by all the droplets during routing is preferred to be minimized to achieve better fault tolerance and robustness, because cells may be defective due to manufacturing issues. Finally, after the solution (routes of droplets) is found, it can be stored into the DMFB to perform pre-programmed biochemical operations. Hence droplet routing is a fundamental design step and crucial to the reconfigurability of DMFB. [1] In this thesis, we propose an elegant routing method that can directly address the electrode interference

[1] Some other works will consider cell contamination and try to make routing path disjoint so as to avoid the residue left on the cross site corrupts the latter passing droplets. This can be addressed in our framework by modifying the weight function to discourage cell reuse, or by representing the paths of routed nets as blockages.

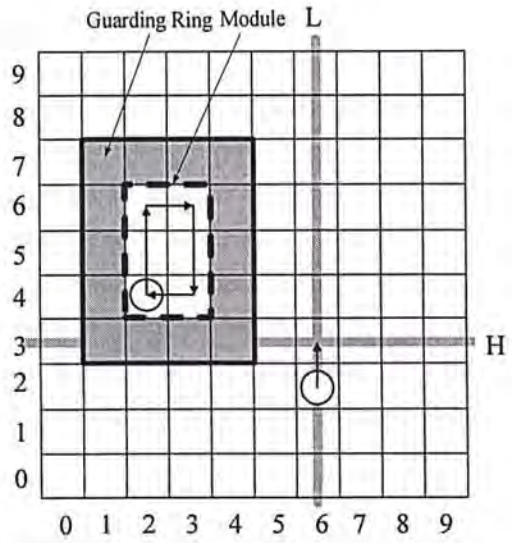


Figure 1.8: A droplet is inside a mixing module and a droplet is moving upwards. The gray rectangle denotes the guarding ring of the module. Note that low voltage cannot be assigned to column 1-4, otherwise, cells in the guarding ring will be extra-activated.

issue and solve the problem.

Note that the blocks in droplet routing is different from the blocks in traditional routing. In fact, they are *on-going operations*. Due to the reconfigurability of DMFB, these operations can be done anywhere on the biochip. In order not to violate the fluidic constraint [5], a *guarding ring* is imposed around a module with on-going operation. During routing, if the droplet inside a module is pulled outside unintentionally, the correctness of the whole bioassay will be ruined. Hence, it is very important to ensure that the droplet will stay inside a module when the operation is still in progress. However, as introduced in Section 1.2.3.1, there may be extra-activated cell appearing on the biochip during droplet routing. They are *allowed to exist* if and only if no electrode interference will happen. Nevertheless, previous works on routing did not consider this potential problem. In this thesis, we forbid the cells in the guarding ring to be activated in order to prevent the droplet from being pulled out. But the cells inside a module are allowed to be activated because the droplet inside will not be pulled out in any way. An example is illustrated in Figure 1.8.

1.6 Our Contributions

In this thesis, a placement method is proposed to solve the placement problem in cross-referencing biochips. The properties of cross-referencing biochip are utilized in the proposed method in order to generate placement result that is more suitable for routing. Meanwhile, we propose a method called CrossRouter that directly solves the droplet routing problem on cross-referencing biochips. Our goal is to simultaneously move a group of droplets as much as possible to maximize the parallelism and minimize the total transportation time, as well as minimizing the total number of cells used during the routing process. The major contributions of our work includes:

- We design a router that directly solves the droplet routing problem in cross-referencing biochips. An elegant two-coloring approach is used in our router to handle the stringent electrode constraints. The experimental results show that the proposed router can achieve shorter average routing time and satisfies timing constraints in all benchmarks in comparison with the latest work. Moreover, the number of cells used in the solution generated by our router is close to [6], although [6] is working on direct-addressing biochips which does not have those stringent electrode constraints. Furthermore, the extra-activated cells around the guarding ring of a module (see Section 1.5) is handled in our router to avoid unwanted effects. Finally, rip-up and re-route process is designed for the router. A probability-based scheme is used to allow non-deterministic ripping, which can break infinite looping effectively.
- We propose a ILP-based method that solves the placement problem in cross-referencing biochip. Our method considers the property of cross-referencing and is droplet routing aware. The effectiveness of our method is verified by running our proposed router on the placement result generated by our method and that by the previous work, respectively. The experimental results suggest that our method can generate placement that is more suitable for a cross-referencing biochip router. To tackle the complexity of the ILP, a partitioning method is included. A maze routing is utilized in our pin assignment stage to eliminate unroutable solutions. Routability can be improved significantly with these extensions. By combining all these efforts, more benchmarks are solved and new experimental result is available.

1.7 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 presents a thorough study of the existing works on the droplet routing problem and placement problem. In Chapter 3, the proposed router for cross-referencing biochip is introduced. Chapter 4 presents the proposed ILP-based method of solving the placement problem in cross-referencing biochips. Note that the droplet routing problem and our solution will be discussed before the placement problem and our approach, although the placement stage is before the routing stage in the physical design of biochip. This is for the purpose of better explaining the problem since some important concepts and preliminaries will be better understood in the context of droplet routing problem. Finally, a conclusion of our work is drawn in Chapter 5.

□ End of chapter.

Chapter 2

Literature Review

2.1 Introduction

Computer-aided design problems of Digital Microfluidic Biochip (DMFB) is receiving much attention in recent years. Placement problem is a key problem in the synthesis of biochip because it directly affects the routability of the droplets. It refers to the placement of microfluidic modules such as mixers and storage units on a biochip. The reconfigurability of digital microfluidic biochips enables various modules to be placed at the same place in different time spans. Routing problem is the last step in the physical design of biochips. All the droplets on the biochip must be routed to their sinks within the timing limits and without unexpected mixing. To ensure a high-throughput, the routing must be done efficiently to achieve a small average routing time. Note that both of the placement and routing problems on DMFB are proved to be NP-hard [23, 24]. In this chapter, we review previous works on both problems. The rest of the chapter is organized as follows. Section 2.2 introduces previous works on the placement problem for biochips, while the previous methods on the droplet routing problem for biochips are reviewed in Section 2.3. Concluding remarks will be given in Section 2.4.

2.2 Previous Works on Placement

There are several papers on solving the placement problem in biochips [25, 26, 27, 28, 29, 7]. But so far, none of them focus on cross-referencing biochip. All of these methods are based on simulated annealing that search for an optimized solution. Based on the representation of the placement configurations used, i.e., direct representation or encoded representation, these methods can be classified into two categories. The following sections review these methods.

2.2.1 Basic Simulated Annealing

The first work for the placement problem in biochips is presented in [25, 27]. Given the scheduling result, a simulated annealing-based module placement algorithm is proposed to solve the problem. A direct representation of the placement configurations is used. The initial placement configuration is constructed by simply joining one module's upper right corner to the next module's lower left corner. New placements are generated by (1) randomly move one module; (2) randomly move one module and change its orientation; (3) switch a pair of modules; (4) switch a pair of modules and change at least one of its orientations. A controlling window for each module is set up to prevent sharp increases in the cost metric due to long-distance moves. The cost metric consists of the area of the array and the degree of fault tolerance. To evaluate the fault tolerance ability of a biochip, they proposed a term *fault tolerance index*, which is computed as the ratio between the number of fault-free cells and the total cell number.

Remarks: The papers [25, 27] solved the placement problem by using the vastly-used simulated annealing approach. However, it does not mention clearly the fall-back strategy when the layout area cannot contain all the modules in the initial placement. Besides, only one bio-assay test case is used in the experiment, which cannot demonstrate the robustness of the proposed method. Lastly, the effect of the fault tolerance index is not discussed or studied.

2.2.2 Unified Synthesis Approach

Overview: A unified synthesis approach is proposed by Su *et al.* in [26], in which placement is one of the steps. In their paper, a *parallel recombinative simulated annealing* (PRSA) method that combines scheduling, resource binding and placement together is proposed. This class of algorithms is best viewed as genetic algorithms that use *Boltzmann trials* between modified and existing solutions to select the solutions that exist in the next generation [26]. Design specification, the module library and a sequence graph modeled from the protocol of the bioassay will be given as input to the algorithm.

Representation and Operation in the Annealing: *Random key*[1] is used as the representation of chromosomes, where the *resource binding of operations*, *delay time of operations* and *operation priorities* are encoded as genes in the chromosomes. The *Reproduction* operation simply selects the best solution to the next generation. The *Crossover* operation uses a parameterized uniform crossover to generate child chromosome from two randomly selected parent chromosomes. Finally, the *Mutation* operation randomly generates new chromosomes.

Construction Procedure and Fitness Function: A Construction Procedure is used to evaluate the fitness value of a chromosome. It consists of three phases. The first phase is *resource binding phase*. The reconfigurable and non-reconfigurable resources are assigned to the operations according to their associated gene value in a chromosome. The duration times for the corresponding operation are then obtained after the first phase. Note that resource constraints are temporarily ignored in this step. In the second phase, i.e., scheduling, the actual start time of each operation can be computed by summing up the operation durations of its predecessors and the associated delay values from the genes in a chromosome. After this phase, a scheduled sequencing graph with resource binding is obtained. The last phase is a placement phase. A greedy algorithm is used in this phase. Microfluidic modules are first sorted in a descending order of their priority values that are represented in a chromosome. In each step, the module with the highest priority among the unplaced ones is selected and placed. An available location for this module is sought while minimizing the array area. Finally, a fitness value can be evaluated for the

[1] A random key is a random number sampled from $[0, 1]$.

candidate chromosome, and it is computed as $\alpha \times A/A_{max} + (1 - \alpha) \times T/T_{max}$, where α is a manually set weight, A is the solution area, A_{max} is the chip size, T is the completion time of the solution and T_{max} is a constant for normalization.

Remarks: This paper proposed a novel unified synthesis method optimizing multiple objectives. Nevertheless, the greedy placement algorithm is not explained in detail in the paper. Furthermore, their objective function contains only the final size of the placement and the assay completion time, while the last step in biochip design, droplet routing, is not considered. Hence, the solution may potentially have a low routability. Feasible pathway of a droplet is not guaranteed to be found, especially in a congested or small-size biochip.

2.2.3 Droplet-Routing-Aware Unified Synthesis Approach

Xu *et al.* extended the PRSA (genetic algorithm) method in [26] by considering the droplet routing to increase routability during the synthesis step [28]. They claimed that the synthesis and placement of DMFB is very similar to the operation of dynamically reconfigurable FPGAs (DRFPGAs) [30]. To be droplet-routing-aware, routability of the placement solution is evaluated by adding estimations of the maximum and average routing length of the chromosome in the PRSA method introduced in the previous section. Since a droplet is only routed between two interdependent modules, e.g., routing a mixed droplet to an optical detector, the routing length of a droplet is modeled by computing the distance between these two modules. This guarantees that the routing complexity is reduced to a certain extent.

Remarks: The proposed method considers droplet-routing in the simulated annealing process. It is shown that the proposed method can generate placement results that have shorter assay completion times, compared to those generated by routing-oblivious methods. However, similar drawbacks still exist as in the aforementioned unified approach. In addition, too many objectives are stuffed into a single objective function, i.e., maximum completion time, size of layout area, fault tolerance and routability. The behavior of the algorithm becomes random and unpredictable.

2.2.4 Simulated Annealing Using T-tree Representation

Yuh *et al.* adopted the T-tree formulation to solve the placement problem [7]. They discussed different several representations for simulated annealing-based methods for the floorplanning problem, including *3D-subTCG* [31] and *T-tree* [32], and showed the advantages of T-tree over those representations. The placement problem is modeled as a *three-dimensional floorplanning problem* and is solved using simulated annealing with the T-tree representation.

3D Temporal Floorplanning and T-tree Representation: In 3D temporal floorplanning, each task is a module that can be placed in a 3D space, where X and Y are the axis in two geometric directions, and the T-axis represents time. T-trees are 3-ary trees, where each node corresponds to a unique task and has at most three children to represent the dimensional relationships among tasks. Given a set of m tasks, let W_i , H_i , and T_i denote the width, height, and duration of each task, where $1 \leq i \leq m$. The coordinates (x_i, y_i) and (x'_i, y'_i) denote the coordinates of the bottom-left and top-right corners of a task v_i respectively. Let t_i and t'_i be the starting time and finish time of v_i, for $1 \leq i \leq m$. A T-tree represents the geometric relationships between two tasks as follows.

- If node n_j is the left child of node n_i, module v_j must be placed adjacent to module v_i in the $T+$ direction, i.e., $t_j = t_i + T_i$.
- If node n_k is the middle child of node n_i , module v_k must be placed in the $Y+$ direction of module v_i, i.e., $y_k \geq y_i + H_i$ and $t_k = t_i$.
- If node n_l is the right child of node n_i , module v_l must be placed in the $X+$ direction of module v_i, i.e., $t_l = t_i$ and $y_l = y_i$.

A T-tree can represent a compacted placement. For each task in a biochip sequence graph, a node is created in a T-tree. A 3D box is modeled for each reconfigurable operation such as mixing, and a 3D box with zero height is modeled for the tasks that takes no time to finish, i.e., dispensing, etc. Note that for a droplet routed between two interdependent modules i and j such that the droplet moves from i to j, it may not be routed to j immediately after i is finished since the scheduled place for j may still be occupied by some other modules. A *storage module* is then introduced to stall the droplet a period of time. In the T-tree representation, this storage module is modeled as a node with unspecified duration of time, since

the starting and ending time of this storage module depend on the interdependent modules i and j. This storage module node is then inserted between modules i and j. Other constraints such as timing and optical detector binding, etc. are modeled accordingly as in the design specification.

Simulated Annealing (SA) Process: An initial solution is obtained by performing a clustering algorithm. Given a sequence graph, the algorithm recursively merges the storage operations with its corresponding preceding and succeeding modules without violating any constraints. A more compact 3D floorplan can then be obtained, since storage units are removed as much as possible and operations can start as soon as possible. Perturbations include traditional moves, swap and rotation. In addition, a *rebind* operation is introduced. In this operation, a task is randomly binded to another resource unit. Feasibility will be checked after perturbation, and a tree reconstruction is performed to fix violated constraints. The fitness value of a T-tree is evaluated using a formula defined as follows:

$$\Phi = \alpha V/V_{norm} + \beta S/S_{norm} + \gamma M \tag{2.1}$$

where V is the volume of the 3D floorplan, S is the sum of the volumes of all the storage units, V_{norm} is the normalized volume, S_{norm} is the normalized sum of the volumes of all storage units, and M is the penalty term defined as the sum of the normalized excessive width, height and assay completion time of the 3D floorplan.

Remarks: This paper maps the placement problem to a 3D floorplanning problem and solves it using simulated annealing, where T-tree is used as the placement representation. However, we can see in the cost function that routability is not considered. Only area and the assay completion time are taken into account. Although a very compact placement solution can be generated, the droplet may not be able to reach the sinks in time and thus the bio-assay cannot be completed.

2.3 Previous Works on Routing

In this section, previous work for biochip routing are classified and discussed according to the electrode manipulations method of biochips, i.e., direct-addressing and cross-referencing.

2.3.1 Direct-Addressing Droplet Routing

Many early works assume direct-addressing scheme when solving the biochip routing problem. This section reviews previous methods on droplet routing based on direct-addressing scheme.

2.3.1.1 A* Search Method

Overview: The paper [33] proposed algorithms that use *A* search* as a framework to generate the routing solutions. Due to the high complexity of A* algorithm, a prioritized version is proposed in their later papers [34, 35].

Basic A* Search: Let A_t be the configuration of a biochip array at time t, every different A_t is a search node with edges connecting nodes at adjacent times together. Let A_s and A_g denote the start and end configurations. The goal is to find a shortest path from A_t to A_g. The problem then becomes a graph search problem. Two lists, *Open* and *Close*, are maintained. The cost function is defined as $f = g + h$, where g is the Manhattan distance from A_t to the current node, and h is defined as the Manhattan distance between the current node and A_g. Optimal solution is guaranteed to be found using this method, but the search space is exponentially huge. At each time step, each droplet has four possible moves, then for d droplet, the branching factor is 4^d. Suppose there are s steps, there will then be 4^{ds} nodes. Pseudo code can be found in [35].

Prioritized A* Search: The author tries to assign priorities to droplets so as to reduce the problem size with loss of optimality. The proposed algorithm uses either a random order or an application specific order as the priority. Obviously, the priority is very important to the quality of the solution. Suppose that each droplet i is assigned a priority $pr(i) = k$. During the branching step, droplets are moved in a decreasing priority order, i.e., $pr(i) = 1$ will move first. Droplets with a higher priority are regarded as moving obstacles for lower priority droplets. Droplets with a lower priority are ignored by droplets with a higher priority.

Remarks: Since the basic A* search is a systematic search technique, optimal solution is guaranteed to be found if there exists. However, it suffers in several aspects. First of all, A* search needs huge computational cost and thus it is limited

to small scale biochip. Furthermore, although the priority assignment is crucial to the routing solution, the author fails to give a reasonable heuristic for it. Lastly, multi-pin nets are not considered in their work.

2.3.1.2 Open Shortest Path First Method

Overview: The authors of [3] proposed a method that combines component layout and droplet routing. The droplet routing problem is modeled as a network routing problem and solved using an algorithm similar to the Open Shortest Path First (OSPF) protocol. In their design, the components (e.g., mixing) will be 'virtually' placed and aligned. Thus, the chip area is logically partitioned into a series of functional areas, and each component is connected by 'street' and 'corners'. Cells of the crossings and streets are then selected as *routers*, and droplets are viewed as *packets* to send. Note that the directions of the receive port and the send port in the routers are specified according to the layout. Hence, the droplet routing problem is now modeled as a network routing problem. Figure 2.1 illustrates the components layout.

OSPF Routing: After the layout is decided, a *component graph* is constructed from components. The component graph is a directed graph in which each node is a component and each edge is a connection between adjacent components. Each intersection (router) contains a routing table, that stores the exits that have the shortest path to the corresponding connected components. The routing tables are initialized using Dijkstra's algorithm. At each clock cycle, the intersections are chosen in a fixed order to select their droplet routing moves. Droplets that have a destination can then be routed towards their destinations, and those that do not have a destination will be assigned one from the routing table. Stalling will be performed if a moving droplet cannot finds a viable exit. Subsequently, the droplets are moved synchronously to their destinations.

Remarks: Modeling droplet routing as a network routing problem and solved using the OSPF protocol is an innovative idea. However, it has several drawbacks during the design. First of all, the scheduling, resource allocation, placement and routing are tightly coupled together, which makes the problem too complicated to be solved. Secondly, the component layout method is too restricted. The droplet

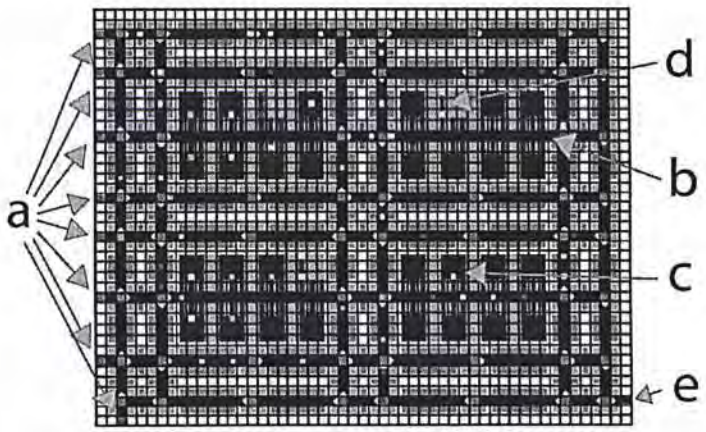


Figure 2.1: Array layout for the PCR analysis from paper [3]. Each cell of the array is represented by a square; arrowheads indicate valid droplet-motion directions. On the left side of the array are (a) eight sources, which supply the input sample droplets to the system. There are (b) four work areas on the array, in which droplets are (c) mixed together and (d) split apart. In the lower right corner of the array is a (e) sink, which moves the droplets of the final products off the array.

routing path is so narrow that only one droplet can be allowed at a time, which inevitably reduces the throughput of the bioassay. The dynamic reconfigurability of digital microfluidics is wasted. Finally, the overhead of initializing and maintaining the routing tables requires a large storage space in the biochip, which may increase the production cost.

2.3.1.3 A Two Phase Algorithm

In the paper [5], a two phase algorithm is proposed.

Phase 1 M-shortest Path: In the first stage, M alternative shortest routes for each droplet will be generated. Lee's algorithm is used to generate the shortest paths for each 2-pin net. For 3-pin nets, it is equivalent to Steiner Minimum Tree problem, and a modified Lee's algorithm with heuristic is applied to find a path. Note that M paths will be generated for each net for selection in the next phase.

Phase 2 Random Selection: In the second stage, a routing path is selected randomly from the corresponding set generated in the first stage for each droplet. The routing paths are combined iteratively to form the final solution. A scheduling approach is used to coordinate the droplets to avoid violation of constraints. The above random selection repeats for a certain number of times and the best solution is chosen from the results. Net-routing-order dependence is addressed to a certain extent.

Droplet Motion Modification: In the first stage, the routes are generated irrespective of the existence of other nets. In order to coordinate the droplets' motions, some droplets may be stalled during transportation. A table of *modification rules* is devised subject to the fluidic constraint. Then, the behavior of a droplet will be modified according to the modification rules in case of collision.

Remarks: The proposed algorithm is simple and straightforward. However, each net is routed regardless of other droplets in the first stage. The potential of collision may be high in the second stage. Although stalling can help to avoid collision, the solution set from the first stage is already fixed and cannot be modified. Hence, the algorithm behaves in a greedy and trial-and-error manner, which may eventually render failure in routing.

2.3.1.4 Network-Flow Based Method

Overview: A two-stage algorithm is proposed in [36]. In the beginning, the *criticality* of each net is calculated. The first stage is global routing. A set of independent nets with the maximum net criticality are identified first. Then, a rough routing path of each net is found. A network-flow algorithm is applied to generate the optimal paths. Detailed routing is done in the second stage. A negotiation-based routing scheme is devised to generate detail paths and schedules.

Net Criticality Calculation: A net is defined as 'critical' if it has fewer possible solutions due to timing constraint, or there are more nets whose solutions affect the solution of this net. The authors defined an equation to calculate the criticality value of a net [36]. The larger the criticality value of a net, the more critical it is.

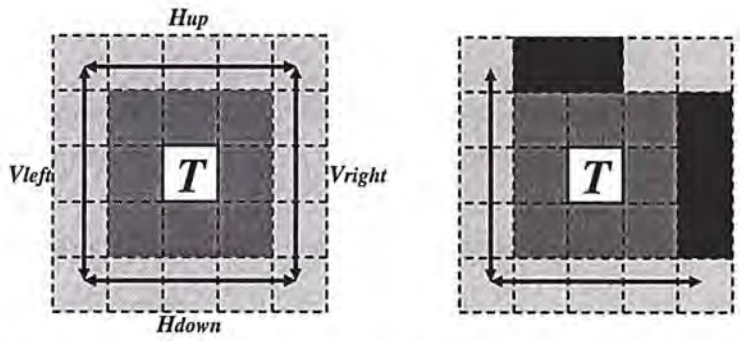
Global Routing: In this stage, a set of *independent nets* that has the maximum sum of criticality is selected. Two nets are referred as independent if they satisfy some criteria [36]. According to the definition of independence, a *conflict graph* G_c is constructed for selection of independent nets. Each node v_a represents a net n_a, and has a weight equal to the criticality value. Two nodes are connected if they are not independent. Then, the net selection problem is reduced to finding the maximum weighted independent set in G_c, which is NP-hard. An $O(N^3)$ heuristic is proposed, where N is the number of nodes in the graph. After the selection of independent nets, the routing problem is formulated as a network-flow problem. To achieve this, a network graph is constructed from the biochip. The biochip is divided into a set of global cells, where each of them contain 3×3 basic cells. Each vertex of the graph corresponds to a global cell. A capacity is associated with a vertex, and is computed as the number of nets whose bounding boxes include this vertex. The capacities of the vertices are then transferred to the capacities of the edges. Finally, a network-flow algorithm is applied to select paths for droplets in the graph.

Detailed Routing: In this stage, the nets from global routing are routed iteratively in a decreasing order of their criticalities. A negotiation-based searching will be performed on the global route to determine the detailed movement steps of each droplet.

Remarks: This paper presents a novel yet complicated approach for the droplet routing problem. The major drawback is that the network flow formulation is significantly affected by the distribution of blockages. If the channel width between blockages is less than three unit cells, it will not be formulated as a global cell in the network flow, and thus the search space is limited.

2.3.1.5 Bypassibility and Concession Method

Overview: The authors of [6] proposed a novel algorithm that considers congestion. Their method relies on two ideas, *bypassibility* and *concession zone*. Bypassibility is used to quantify the degradation of routability after a droplet is routed, while concession zone is introduced to temporarily store droplets to break deadlock. After a sequential schedule of droplet routing is obtained using bypassibility



(a) A 5x5 window is considered to evaluate the bypassibility. Four bypasses are shown right out of the shadowed regions.

(b) This example has full bypassibility, as there exist at least one vertical and one horizontal bypasses.

Figure 2.2: Bypassibility illustration

and concession zone, a *compaction* process is applied to shorten the routing time so that the timing constraint is satisfied.

Routing by Bypassibility: Bypassibility is defined by the free bypass way surrounding the 3×3 region occupied by a droplet after it is routed. An example is given in Figure 2.2. The author claimed that droplets with higher bypassibility and have paths to their destinations should be routed earlier, since there are still ways for other droplets. At first, the bypassibility of each droplet after routing is computed. Then, the droplets are routed sequentially in a decreasing order of bypassibility. The starting time T_b of a routing is the finishing time of the previously routed droplet.

Routing with Concession: Deadlock may appear during the naive sequential routing described in the last paragraph. Concession zone is proposed to solve the problem. It is defined as an unoccupied space in the chip which is larger than a 3×1 window. When deadlock happens, a droplet d_i with the longest distance to a concession zone is routed first. The blocking droplets should be moved to a concession zone in order to give way to d_i. Before the droplet d_i can move, it should stall at its original position for some amount of time α_i that is used for the blocking droplets to move to the nearest concession zones. The value of α_i can be iteratively computed by finding the blocking droplets and summing up the total

time they need to back off.

Solution Compaction: Since the routing solution is generated sequentially, the timing constraint may be violated. In this stage, all droplets including those unrouted one are rerouted greedily to compact the solution. In particular, all nets are routed repeatedly to satisfy the timing constraint until no improvement can be made.

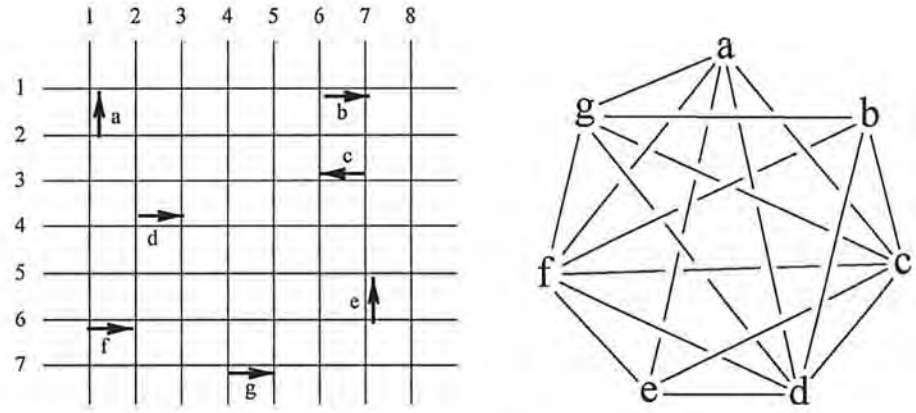
Remarks: The proposed method achieves the best routability compared to all the previous methods. However, it has several issues not addressed. The proposed bypassibility only considers the congested status after a droplet is arrived at its sink. It cannot reflect the dynamic changing status during routing. The concession zone, however, seems to solve this problem. But the droplet that is moved to a concession zone may recursively blocked or be blocked by other droplets. Finally, the solution compaction uses a greedy method, which cannot guarantee to find a correct ordering of the routes and may lead to a timing violation. Finally, 3-pin nets are not mentioned in the paper.

2.3.2 Cross-Referencing Droplet Routing

Existing methods of solving the cross-referencing droplet routing problem can be classified into two categories. The first category is to get a direct-addressing solution first, and then convert it to satisfy the electrode constraint. This is done by splitting each *original* time step in the direct-addressing solutions into a number of *real* time steps in the cross-referencing biochips. Since there are many existing methods for direct-addressing droplet routing, it is reasonable to adapt them for cross-referencing biochips. Nonetheless, this kind of approach might be myopic since it loses the global view of the problem. The second category is to solve the problem directly, which is not limited by the direct-addressing result right from the beginning.

2.3.2.1 Graph Coloring Method

Griffith *et al.* tackles the problem using a graph coloring approach [3]. A solution for direct-addressing biochips is obtained first by using the method proposed in



(a) Grid of control wires indicating droplets with horizontal and vertical movements.

(b) Corresponding transition graph for droplet movements.

Figure 2.3: An example of the transitive graph [3].

paper [37]. Then, the solution is converted to a cross-referencing one. The idea is to find sets of droplets that can be moved simultaneously without violating the fluidic constraint nor causing electrode interference. Then, each time step in the original direct-addressing solution is split to one or more real clock cycles.

Graph Construction and Coloring Algorithm: A *transitive graph* is constructed for each time step of the original routing solution. The vertex set of the graph is the set of all movements that must be performed during an original time step. The edge set consists of all pairs (u, v), where u and v are two vertices such that their corresponding movements cannot be performed in the same real time step. In other words, each edge represents a moving constraint. An example is shown in Figure 2.3. The chromatic number is the smallest number of real time step needed to implement this original move. It is proved that the chromatic number is equal to the number of maximum independent sets. An $O(|V|^3)$ heuristic-based algorithm is proposed to find sets of independent vertices iteratively. Starting from the original graph, an independent set is found by randomly selecting the vertices that are independent. This set of independent vertices and the corresponding edges are then removed from the graph. This process continues for the remaining graph until it becomes empty.

Remarks: The proposed algorithm solves the problem by splitting each time step in the direct-addressing solution into several time steps that will not cause any electrode interference in the cross-referencing biochips. Nevertheless, the graph modeling in their paper is so restrictive that the constructed graph is almost a clique. In addition, the heuristic-based algorithm may fail to find a small chromatic number. Both of these two drawbacks tend to make the converted solution nearly sequential. This seriously hampers a high-throughput of the bio-assay.

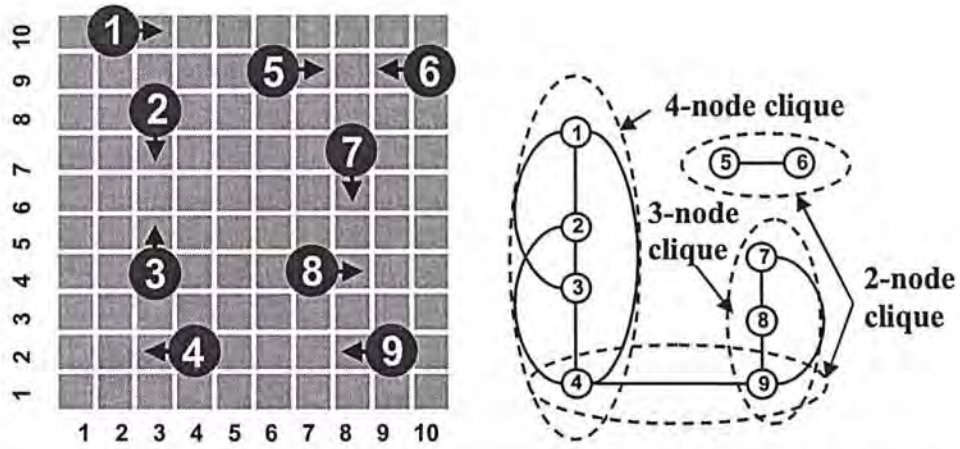
2.3.2.2 Clique Partitioning Method

In the paper [4], a clique partitioning approach is proposed to adapt the solution generated for direct-addressing biochips to a cross-referencing one. The direct-addressing solution is based on the method proposed in [5]. Similar to the graph coloring method in Section 2.3.2.1, their approach also splits the original time step into several time steps in cross-referencing biochips. At each time step, the droplets that share the same column (or row) in their next time steps are categorized as a group. These droplets can be moved simultaneously without causing any electrode interference because no extra cells will ever be activated. Then, groups are found iteratively until all droplets are moved. Hence, the total completion time for a set of droplets is determined by the number of groups categorized.

Graph Model and Clique Partitioning: Based on the categorization method mentioned above, a *droplet movement graph* is constructed for each time-step. An algorithm is utilized to find a minimum clique partitioning, which is a NP-hard problem. [2] A heuristic is proposed to iteratively find the cliques. The largest clique is first determined, and then the corresponding vertices and edges are deleted from the graph. The algorithm terminates when all the nodes in the graph have been deleted. See Figure 2.4 for an example.

Remarks: The computational complexity of the proposed method is small. However, the categorization of the droplets are far from efficient. Only a few droplets can be moved at a time. In the worst case, i.e., no droplets share the same row or column in the destination cells, the resulting solution becomes sequential for this

[2]The original paper uses 'minimal clique partition', which is incorrect.



(a) Example to illustrate destination-cell-based categorization.

(b) Mapping destination cell layout to undirected graph and find for clique partition.

Figure 2.4: An example of the proposed clique partitioning method [4].

time step. This case is commonly seen in large bio-assays, since droplets are far away from each other.

2.3.2.3 Progressive-ILP Method

Yuh *et al.* proposed a state-of-the-art integer linear programming (ILP) based method to solve the droplet routing problem on cross-referencing biochips directly [20]. They claim that it is the first method that directly solves the problem, not relying on a direct-addressing result. The fluidic constraint, electrode constraint and timing constraint are modeled as ILP constraints. The latest arrival time is used as the objective function. Then, the movements of droplets are determined by solving the ILP. However, due to the high complexity of solving the ILP, they proposed a progressive ILP method, which iteratively determine the movement of droplets at each time step by solving an ILP. A 'droplet movement cost' is used as a metric to evaluate congestion when solving the ILP. After the division of problem, the progressive version of ILP can be solved in a time-efficient manner.

Remarks: This method works directly on cross-referencing biochips, and hence better result can be obtained. However, their approach cannot guarantee to finish

all the routings within a given timing constraint and this might eventually result in failure of the whole bio-application. Moreover, the number of cells used is not taken into account, which is an important objective that need to be considered in droplet routing.

2.4 Conclusion

In this chapter, previous works on the placement and droplet routing problems are discussed. For direct-addressing droplet routing, A* search method, OSPF-based method, a two-phase method, network-flow based method and a bypassibility method are introduced. For cross-referencing biochips, indirect approaches including graph coloring and clique partitioning approach are introduced. A drawback of these methods is that they both use direct-addressing as a starting point. Hence, the solution quality is potentially affected. More importantly, the constraints of moving multiple droplets modeled in their methods are too limited. The electrode manipulation can be far more flexible. In fact, cells can be extra-activated as long as no problem occurs. A progressive ILP method that directly solves the problem is introduced at last. Their method cannot route all the droplets within the timing limit. In addition, this method does not take cell usage into account. In the next chapter, we will introduce our routing method that addresses all of the above issues.

□ End of chapter.

Chapter 3

CrossRouter for Cross-Referencing Biochip

The content of this chapter is included in the *Proceedings of the 15th Asia and South Pacific Digital Automation Conference (ASP-DAC) 2010* [38].

3.1 Introduction

The droplet routing problem is the last step in the design of biochips, and it is the step before the placement problem. It refers to moving the droplets from their sources to sinks within the timing limits so as to perform the incoming operations in time. During the process, the droplets must not be mixed unexpectedly. Otherwise, the whole bio-assay will be ruined. In contrast to the traditional routing problem in VLSI CAD, this problem is more similar to motion planning problem in robotics research since the droplets do not occupy a certain cell permanently. Originally, only timing constraint and fluidic constraint need to be considered in the droplet routing problem on direct-addressing biochips. The cross-referencing biochip introduces a stringent electrode constraint which may cause unwanted effects and hamper a high-throughput of a bio-assay. Hence, it needs to be handled carefully. In this chapter, we will detail a droplet routing method called CrossRouter for cross-referencing biochips. Our method uses a two-coloring graph approach to elegantly handle the electrode constraint. Experimental results show

that by using our method, cross-referencing biochips can still achieve a high-throughput as in direct-addressing biochips.

The rest of this chapter is organized as follows. A formal formulation of the droplet routing problem will be given in Section 3.2. An overview of our algorithm is given in Section 3.3. Each part of the algorithm will be described from Section 3.4 through Section 3.7. The Experimental result will be given in Section 3.8. Finally, a conclusion will be drawn in Section 3.9.

3.2 Problem Formulation

In this problem, all the droplets on a biochip must be routed to their destinations subject to some constraints. A cross-referencing DMFB can be viewed as a 2D array with W rows and H columns. Each cell on the biochip can be referred to as (X, Y). There are D droplets and B blockages on the biochip. The fluidic ports on the boundary of a module are called *pins*. A droplet route between the pins of different modules are modeled as a *net* [5]. Each droplet is either in a *2-pin net* or a *3-pin net*. There are totally N nets on the biochip. Given a droplet d_i (net i), we denote the start location and the end location of d_i as *source* s_i and *sink* t_i. Usually, there is a *waste reservoir* on the biochip. When a droplet reaches such locations, it will be removed from the biochip at next time step. Finally, a time limit T is given, which is an upper bound on the total amount of time used to route all the nets. Then, the routing problem can be formulated as follows:

- Input:
  - A placement result that contains a list of n nets.
  - Time limit T, chip size $W \times H$, blocks and reservoir location.
- Output:
  - A schedule of voltage assignment at each time step.
- Objective
  - Route all droplets to their destination within timing constraint.
  - Minimize time used and cell used.

3.3 Overview of Our Method

In principle, our method first gets a reasonable routing order, then each net is processed according to this order, while considering the nets which have already been processed. Rip-up and re-route will be performed if a net is unsuccessfully routed. The process continues until all the nets are routed. The solution obtained by our method contains a valid voltage assignment sequence in different time steps, which ensures that each droplet reaches its destination cell within time T without causing any violation. If the time limit for rip-up and re-route is exceeded, the routing process will be stopped, and failure will be reported to the previous synthesis stage, i.e., placement, etc. Tasks should then be re-assigned or module should be replaced in order to avoid over-congested routing configuration.

When routing a net, there are two stages. The first stage is called *propagation*. Our algorithm tries to find a valid shortest path for a net in this stage. The propagation starts from the source of a net, and ends when it reaches the destination. Fluidic and electrode constraint check will be performed in each propagation step. A 3D bitmap technique is used to enable quick detection of fluidic constraint violation, while an incremental two-coloring method is used during electrode constraint check to seek if there exists a feasible voltage assignment to implement the current set of droplet movements. A penalty count for each previously processed net will be incremented by one if it causes violation when routing the current net. These counts are used in the rip-up and re-route step to select the net to be ripped up. The second stage is the *backtracking* stage. After a valid path is found at the propagation stage, voltage assignment is *incrementally* performed based on the routing result. The algorithm is outlined in Algorithm 1. Advantages of our algorithm include the general framework and the low coupling of each part, which provides more flexibility and extensibility in considering new constraints and strategies. Now, we will detail each part of the algorithm as following.

3.4 Net Order Computation

The net order is computed according to the following conditions in descending precedence:

1. If a net i's source point is inside the bounding box of another net j, i should

Algorithm 1: CrossRouter

Input: Configuration of the droplet routing problem.

Output: A valid voltage assignment in each time step.

```
1  begin
2      queue ← compute a routing order of nets;
3      while queue not empty do
4          net i ← pop head element of queue ;
5          while net i not routed successfully do
6              result ← do propagation of net i;
7              if result = SUCCESS then
8                  backtrack from the sink point;
9              else if this is the first net being routed then
10                 report FAIL and exit;
11             else perform rip-up and re-route;
12         end
13     end
14 end
```

be routed before j;

2. Nets with a smaller Manhattan distance from the source point to the sink point should be routed earlier.

3.5 Propagation Stage

In this stage, for a given net d_i, the router tries to find a route $\{d_i^0, d_i^1, \ldots, d_i^T\}$ from its source to its sink without violating any constraint or causing any interference. In contrast to traditional routing, the route of a net here is the temporal positions of a droplet within the timing constraint, and is subject to the control of the electrodes.

Here we present how the possible routes of a net are explored in CrossRouter.

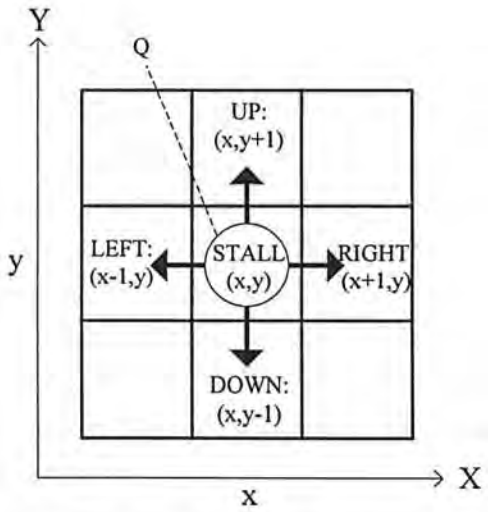


Figure 3.1: Five possible movements of a droplet, where Q is the current position at time t.

During the propagation, a droplet d_i may reach a cell $Q = (Q_x, Q_y)$ at time $t-1$ and move to one of its adjacent cells P at time t. The pair (Q, t), called a *droplet movement status*, denotes the position of the droplet at time t. $(Q, t-1)$ is called the *parent* of (P, t) and is denoted by $(Q, t-1) = parent((P, t))$. There are five possible movements for the droplet at time t. They are *LEFT, RIGHT, UP, DOWN, STALL*, as illustrated in Figure 3.1. For example, when droplet is at $(3, 6)$ at time=3, then at time=4, it can be moved to either one of $(2, 6), (4, 6), (3, 7), (3, 5)$, or stall at the same position $(3, 6)$. However, the movement that returns back to the parent cell is not considered in CrossRouter. This is because if the droplet stalls at $parent((Q, t))$ at time $t-1$, the same result can be obtained but possibly introducing less electrode activation. Hence actually only four movements are considered at each time step, except for the source point at time=0.

By following the parents of a status (P, t) all the way back, one can always trace out the route by which d_i is transported to P. Starting from a source point, a set of statuses are explored iteratively until the sink point is discovered. In order to find a desired path earlier, a *weight* is assigned to each status which is calculated as follows:

$$weight(P, t) = t + MD(P, s_i) + U(P) + Len(P, t)$$

where $MD(A, B)$ is the Manhattan distance from A to B, $U(P) = N - \#used$, N

is number of nets and *#used* is the number of nets that used this cell before, and $Len(P,t)$ is the length of the current path from the source point to P at t.

A sorted list of such statuses are maintained to record the routes that have been explored. At each iteration, the status (P,t) with the smallest weight is chosen. If the sink point is reached at time t, fluidic and electrode check should be performed from time $t+1$ to T so as to ensure that this droplet will not block any processed net. Otherwise, new statuses due to propagation from P at time $t+1$ will be added into the list, subject to the constraints that we will discuss in the next section. If the sink point is not reached and the list becomes empty, the routing of this net is failed. Rip-up and re-route will be performed. This propagation step is summarized in Algorithm 2.

The fluidic and electrode constraint can be checked as described in the next two sections.

3.5.1 Fluidic Constraint Check

Fluidic constraint check should be performed in order to prevent unexpected mixing of droplets during their transportation. To be more formal, let $d_i^t = (x_i^t, y_i^t)$ denote the location of droplet i at time t. Note that $s_i = d_i^0$ and $t_i = d_i^T$. The static and dynamic fluidic constraints can be stated as:

$$\begin{array}{llll} |x_i^t - x_j^t| \geq 2 & \textbf{or} & |y_i^t - y_j^t| \geq 2 & \textbf{and} \\ |x_i^t - x_j^{t-1}| \geq 2 & \textbf{or} & |y_i^t - y_j^{t-1}| \geq 2 & \end{array} \tag{3.1}$$

Furthermore, as the routing can be viewed as a 3D routing, in our implementation, we use a 3D bitmap to speed up the check.

3.5.2 Electrode Constraint Check

Electrode constraint check is a crucial part in CrossRouter. We can show that, when there are only two moving droplets, there will always be a valid voltage assignment to move them correctly without causing any electrode interference. An example is illustrated in Figure 1.5. However, conflict may happen among three simultaneously moving droplets, because the conditions to activate and deactivate some cells may be contradictory. In this thesis, we introduce a succinct graph coloring based method to determine whether the simultaneous movements of a

Algorithm 2: Propagation

Input: d_i is the current routing net.

Output: return route if found, else report FAIL

```
1  begin
2      list ← {(s_i, 0)};
3      repeat
4          (Q, t) ← smallest weight element in list;
5          remove (Q, t) from list;
6          if Q = sink t_i then
7              perform fluidic electrode check on (Q, t+1),...,(Q, T);
8              if no processed net is blocked then return route to Q;
9          else if t + 1 ≤ T then
10             // (R, t − 1) = parent(Q, t);
11             foreach neighboring cells P of Q except R do
12                 perform fluidic and electrode check on (P, t + 1);
13                 if no violation happens then list ← list ∪ {(P, t + 1)};
14             end
15         end
16     until list = ∅;
17     return FAIL;
18 end
```

set of droplets are feasible or not, i.e., whether a valid voltage assignment exists to implement the movements of several droplets at the same time. In contrast to the method proposed by Griffith [37], our approach do not attempt to find the chromatic number for a graph, which is NP-hard in general, but rather determines whether a movement is implementable. We will make use of a special type of *constraint graph* to perform this check.

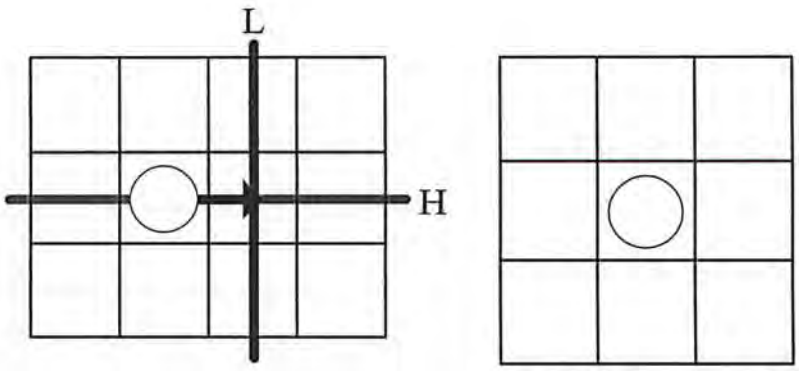
A *constraint graph* $G = (V, E)$ in our context is an undirected graph that

consists of two types of edges. The first kind is called *DIFF* edge, which means that the vertices at its two ends must have different colors, and the second kind is called *SAME* edge, which means that the adjacent vertices should have the same color. A two-coloring in this constraint graph is an assignment of two colors to the vertices such that the vertices sharing a $DIFF$ edge have different colors, while the vertices sharing a $SAME$ edge have the same color. As discussed before, we can apply a high, low or ground voltage to a row or column and a cell will be activated when its intersecting row and column are assigned high and low (or low and high) respectively. In our electrode constraint check, we will have a vertex representing a row or a column in the constraint graph if this row or column must be set to a high or a low voltage in order to activate some cells. For those rows or columns which are not represented by a vertex in the graph, they are supposed to be set to the ground voltage and thus will not cause any electrode interference. (Note that this concept is important allowing us to be able to check the electrode constraint efficiently using two-coloring.)

At time t, a constraint graph G_t will be constructed according to the droplets' movements. We will insert vertices and edges into G_t in such a way that *there is a one-to-one correspondence between two-coloring of the graph and feasible voltage assignment on the biochip.* We will explain below how G_t will be constructed. We use $C(X)$ to denote a vertex in G_t that represents a column X and use $R(Y)$ to denote a vertex representing row Y. During the transition of a droplet from one cell Q to one of its adjacent cells P, only P is allowed to be activated in the droplet's *4 × 3 bounding box (BB)*, which is formed by the neighboring cells of P and Q. Note that the outer row/column perpendicular to the moving direction of the droplets is also forbidden, since its activation may cause unexpected movement of the droplet after it settles in P at time $t + 1$. However, if the droplet is stalling, all the cells in the droplet's 3×3 *BB* cannot be activated, while the droplet's current location Q can be activated or not.

According to the rules above, the constraints can be modeled by adding different types of edges into the graph. First of all, vertices $C(X)$ and $R(Y)$ will be added into G_t if and only if there is a droplet moving to the cell at (X, Y) at time t. We consider the move and stall action separately as follows:

- If a droplet d_i is moving from Q to $P = (P_x, P_y)$, add a $DIFF$ edge between $C(P_x)$ and $R(P_y)$. This is to activate the cell at P. For any neighboring cells (X, Y) of P and Q, if both $R(X)$ and $C(Y)$ exists in G_t (due to d_i or other



(a) 4×3 bounding box of a droplet cannot be activated when it is moving right.

(b) 3×3 bounding box of a droplet cannot be activated when it is stalling.

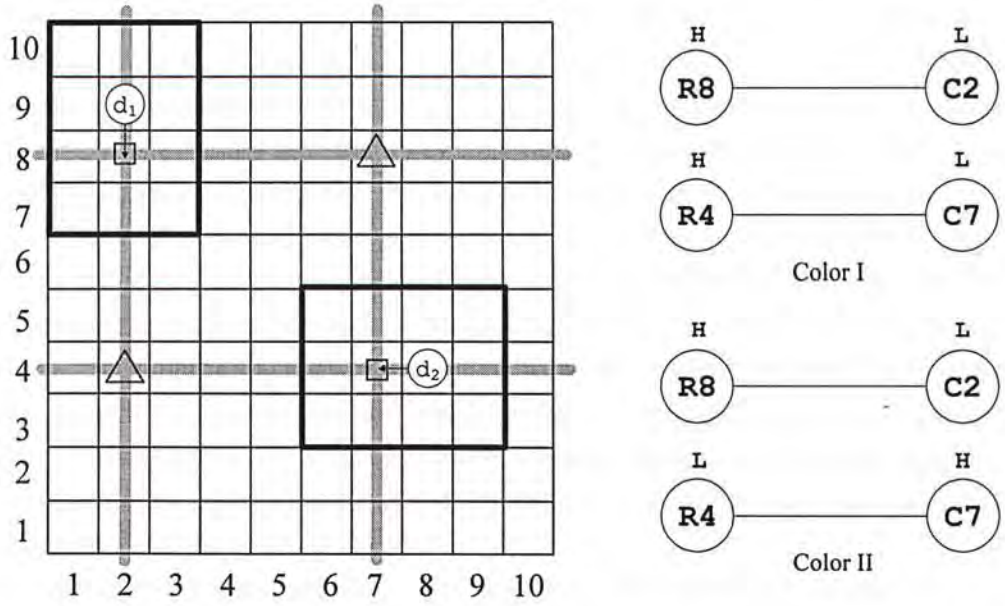
Figure 3.2: Conditions on the movement and stalling of a droplet

droplets), add a $SAME$ edge between $C(X)$ and $R(Y)$. This is to make sure that the neighboring cells are not activated.

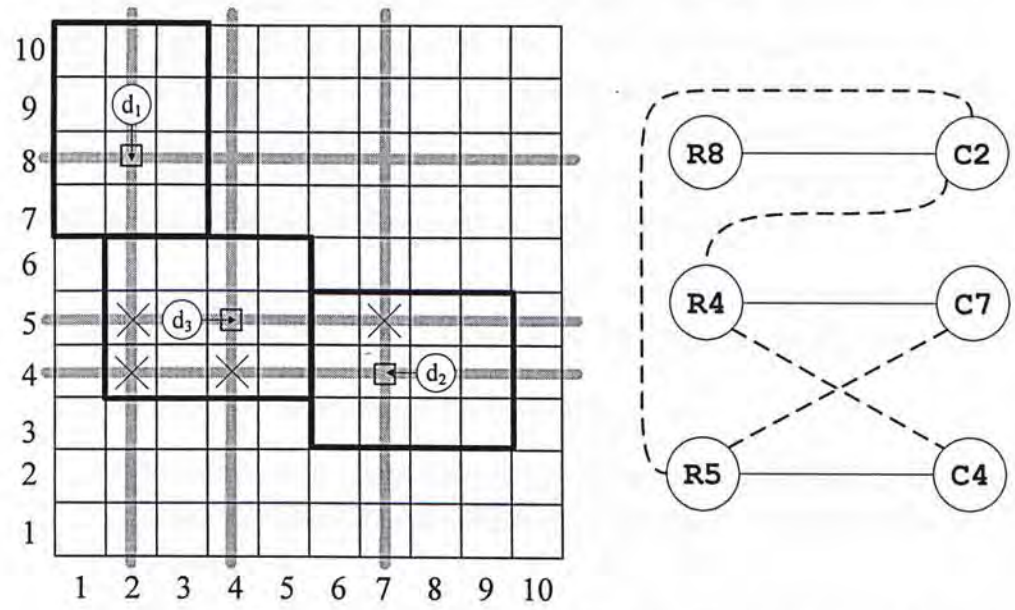
- If a droplet d_i is stalling at its current position P, we consider its neighboring cells (X, Y). If both $R(X)$ and $C(Y)$ exists in G_t (due to other droplets), add a $SAME$ edge between $C(X)$ and $R(Y)$. This is make sure that the neighboring cells are not activated.

After constructing G_t, we can determine whether G_t has a two-coloring, which can be done efficiently. We can easily see that the existence of a two-coloring in G_t is equivalent to having a feasible voltage assignment to the rows and columns such that all droplets' movements can be achieved simultaneously. We illustrate the idea by giving a concrete example in Figure 3.3 to demonstrate the construction and coloring of the constraint graph.

The shaded bars are the rows and columns at which we need to apply high or low voltages. Small rectangles are the cell to be activated. The solid lines in the graphs on the right represent $DIFF$ edges while the dotted lines represent $SAME$ edges. In Figure 3.3(a), we need to activate (2,8) and (7,4), hence $DIFF$ edges are added between $R(8)$ and $C(2)$, and between $R(4)$ and $C(7)$. We can easily see that a two-coloring exists in the constraint graph, and thus a valid voltage assignment exists to bring about the movements of the two droplets. Figure 3.3(b) considers droplet d_3 in addition to d_1 and d_2, new edges are added into the constraint graph.



(a) Chip scenario and constraint graph considering d_1 and d_2 only. Two different two-coloring solution are showed while color I activates more cells (marked with 'Δ')



(b) Chip scenario and constraint graph considering d_1, d_2 and d_3 simultaneously.

Figure 3.3: Example of checking electrode interference with constraint graph

The cells labeled with an 'X' (cells (2,5), (2,4), (4,4) and (7,5)) are those that their row and column vertices exist in G_t, i.e., we need to assign a high or low voltage to them, but we want to make sure that they will not be activated. Therefore, SAME edges will be added between their row and column vertices. For this constraint graph, we can easily determine that no two-coloring exists, and thus there is no valid voltage assignment such that the three droplets can move simultaneously.

Note that this constraint graph seem to be similar to the interference graph with coalescing vertices in the *register allocation* problem. It is known that N-coloring on this type of graphs is NP-hard when $N \geq 3$, while the corresponding two-coloring problem is polynomial-time solvable. Adding the SAME edge constraint will not increase the problem difficulty, *i.e.*, two-coloring of the proposed constraint graph is still polynomial-time solvable. Hence, we can efficiently determine if the current movement of a droplet is feasible by using the proposed method.

In order to avoid constructing the graph from scratch for every electrode check, we implemented an *incremental* two-coloring in which the two-coloring results (with colors assigned to the vertices) are kept for later use. For each time step t, we keep a colored constraint graph G_t. These graphs will be updated and saved whenever a new droplet is successfully routed. When we consider the routing of the next droplet d_j at time t, we will reload the stored constraint graph G_t that contains the vertices and edges due to the movements of droplets $d_1, d_2, \ldots, d_{j-1}$ (assume that we process them in this order) at time t. When an edge $e = (u, v)$ is added due to this droplet d_j, we can check its *feasibility* as follows:

1. If an edge exists between u and v already, we check the compatibility of e with the existing one. Report failure if the two edges are of different types.

2. If no edge exist between u and v originally,

   (a) If either u or v has no color and its degree is 0 before adding the edge e, we can safely color it according to the type of e and the color of the other vertex.

   (b) If the type of e is compatible with the color of u and v, e.g., a $SAME$ edge and both u and v are HI, e can be safely added.

   (c) If the type of e is not compatible with the colors of u and v, we will try to *flip* the colors of all the vertices in u's (or v's) connected component.

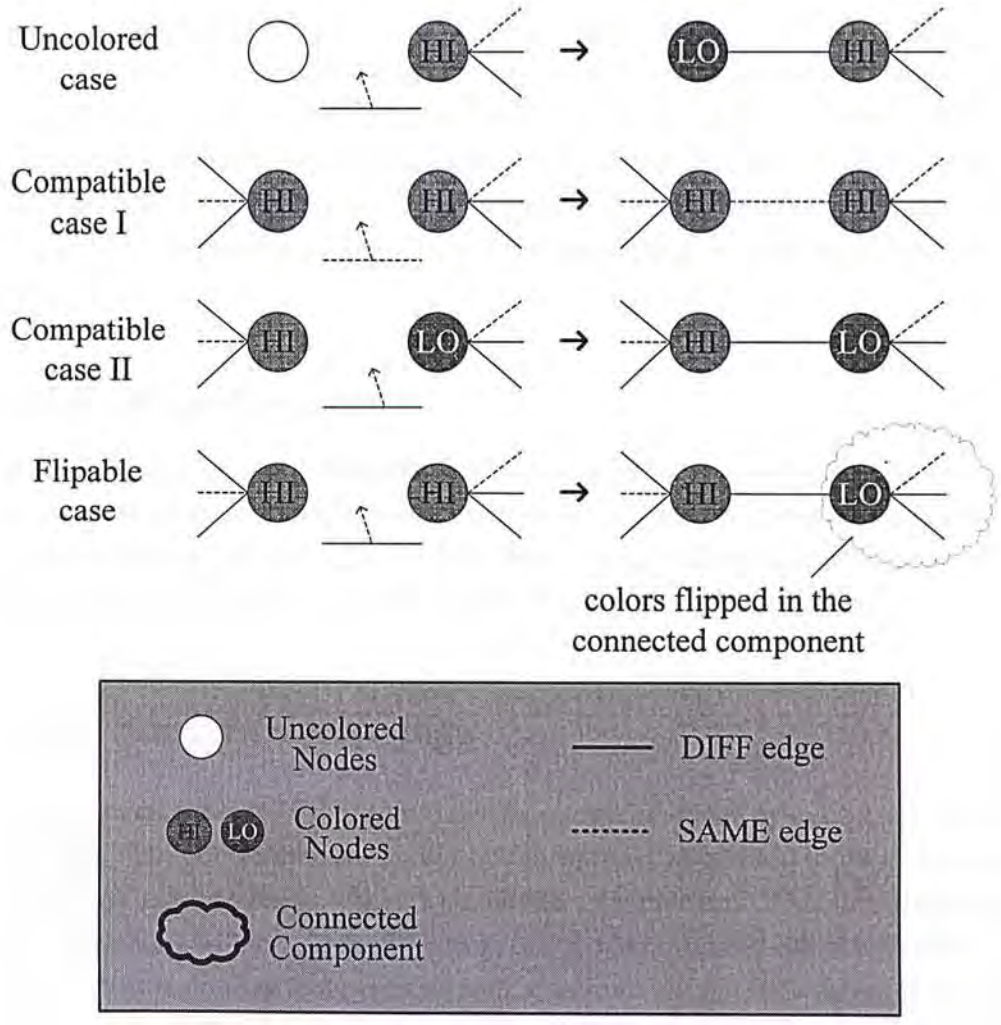


Figure 3.4: Four cases of adding edge in incremental two-coloring.

If the color of v (u) does not change after this flip operation, e can be safely added, otherwise the two-coloring is infeasible.

These four cases are summarized in Figure 3.4.

3.5.3 Handling 3-pin net

Some nets may consist of more than one droplets. This kind of net is called 3-pin net, which models the operation of merging. In our method, each subnet is treated as a 2-pin net and routed to its sink separately. The droplet with a smaller

Manhattan distance from the sink will be routed to the sink point first, and follows the other droplet. Note that they are only allowed to be merged when both moving horizontally or vertically, but not one horizontally and one vertically More specifically, when the two droplets are in the same row or a same column and are separated by one cell, they will both be moved to the middle cell and merged. After the two droplets merge, they become one droplet that will then be routed to the sink point.

3.5.4 Waste Reservoir

Some droplets may have the same destination, because they are to be disposed at a waste disposal location, namely *waste reservoir*. However, we will not allow merging during the routing of such droplets. They will be routed to the waste reservoir and be removed from the biochip immediately it reaches there.

3.6 Backtracking Stage

At this stage, the path of the current routing net will be found by tracing back from the sink. The constraints that caused by this net will be updated to the constraint graphs. If a droplet reaches its sink point before the time limit T. It should occupy the destination cell and serve as a blockage for other droplets. Suppose it reaches its sink at time t, then in the routing path of this net, the droplet's location should be at this sink point from time $t + 1$ to T. Nevertheless, if this net's sink point is a waste reservoir, this droplet will be removed from the biochip at time $t + 1$ and thus no long be an obstacle to other droplets.

3.7 Rip-up and Re-route Nets

During the routing, for each net, the number of conflict it contributes to the current routing net will be recorded. When a valid path cannot be found for the current net, rip-up and re-route will be performed. The conflict counts will then serve as probabilities to determine which net to be ripped. The reason is that the larger conflict count a net has, the more possible that after this net is ripped, the current net can be successfully routed. More importantly, other nets with smaller conflict

Table 3.1: Comparison between Progressive ILP and CrossRouter

Circuit	# sub.[a]	Progressive ILP		CrossRouter		Improvement	
		Max/Avg cycle	CPU (s)	Max/Avg cycle	CPU (s)	Avg (%)	CPU (%)
In-vitro_1	11	24/13.09	2.55	20/12.09	0.92	8	64
In-vitro_2	15	22/11.00	2.53	19/10.73	1.21	2	52
Protein_1	64	26/16.15	15.36	20/15.52	7.76	4	49
Protein_2	78	26/10.23	6.70	20/9.86	2.22	4	69

[a]Number of subproblems in a benchmark.

counts still have chance to be ripped. This randomness can effectively break the tie when some nets are ripping each other as a loop. Note that we start the conflict count from one in order to give every net a chance to be ripped.

3.8 Experimental Results

We use real-life bio-assays as benchmarks. There are four sets of benchmarks, namely *in-vitro*, *in-vitro_2*, *protein* and *protein_2*, and there are 11, 15, 64 and 78 sub-problems in each benchmark, respectively. The *timing constraint* for every subproblem in each benchmark is 20 time units. To evaluate the proposed routing method, we will compare with the state-of-the art work in paper [20]. Both of our and their programs are implemented in C++. Their program is executed on a 1.2GHz SUN Blade-2000 machine with 8GB memory, while our CrossRouter is executed on an Intel 1.6GHz machine with 1.5GB memory.

Table 3.1 gives the comparison between progressive ILP [20] and CrossRouter. The max cycle in Table 3.1 stands for the time spent for routing the subproblem that takes longest time to finish. And the avg cycle stands for the average time to route all the subproblem in a benchmark. The result shows that the routability of CrossRouter is better, since it can route all the subproblems within the timing constraint, while the approach in [20] has its maximum routing time exceeding the timing constraint. For instance, in the benchmark *in-vitro_1*, their router got a finishing time of 24 for some subproblems while ours are within the time limit

Table 3.2: Comparison between [6] and CrossRouter

Circuit	Size	# Cells used	
		HPDRA[a]	CrossRouter
In-vitro_1	16 × 16	258	246
In-vitro_2	14 × 14	246	254
Protein_1	21 × 21	1688	1668
Protein_2	13 × 13	963	976

[a]High Performance Droplet Routing Algorithm for direct-addressing DMFB proposed in [6].

20. Better result on average cycle time and better CPU runtime are obtained, which demonstrates the good quality of our solutions. Since the paper [20] does do not optimize the number of cells used in [20], we made another comparison with HPDRA [6], which is a droplet router for direct addressing biochips. Note that in direct addressing biochips, each cell can be activated independently, so there is no electrode interference and the droplet routing problem is much simpler. Table 3.2 shows that although the constraints are much harsher in the problem we are dealing with, our router can get better result in terms of the number of cells used among the four benchmarks, except for *In-vitro_2*. The reason is that in subproblem 2 of this benchmark, their router found an earlier merging point of a 3-pin net than ours. Figure 3.5 is the routing result at time zero for the 8^{th} sub-problem of the *protein_1* benchmark generated by CrossRouter. We can see that 6 droplet are moving at the same time. Many extra cells are activated, but no electrode interference has ever been caused.

3.9 Conclusion

In this chapter, we have proposed a systematic routing method to solve the droplet routing problem on cross-referencing digital microfluidic biochip. Our goal is to route all the droplets within the timing constraint while satisfying the fluidic and electrode constraints. We first formulate the droplet routing problem on cross-referencing DMFB. Then, the nets are routed using the proposed routing algorithm. The stringent electrode interference is avoided using an elegant two-coloring

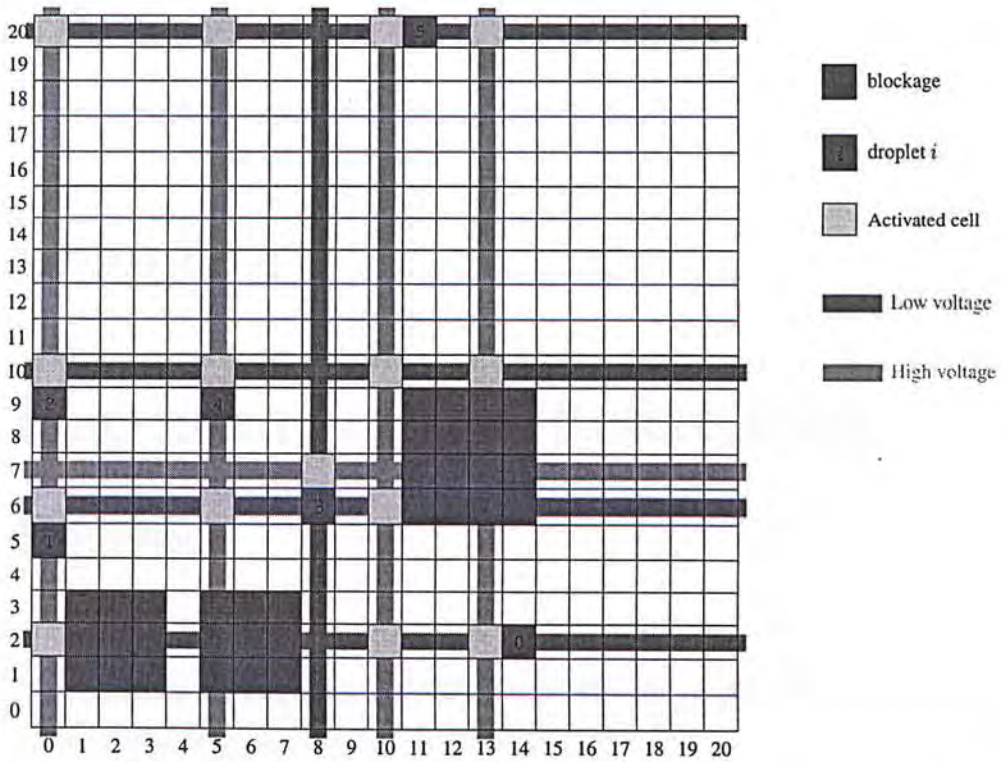


Figure 3.5: Routing solution from the 8^{th} sub-problem of *protein_1* benchmark.

method. Rip-up and re-route will be performed if the routing of current net is failed, or the algorithm detects a deadlock. Real-life benchmarks are used to evaluate our method. Experimental results illustrates the advantages of our method. Compared with previous work, our router improves averagely 4% in routing time and 58% in runtime. It can route all the benchmarks within the time limits, while the latest work fails at some cases.

□ **End of chapter.**

Chapter 4

Placement in Cross-Referencing Biochip

The content of this chapter is included in the *Proceedings of the 19th International Symposium on Physical Design (ISPD) 2010* [39].

4.1 Introduction

Placement is the step before the droplet routing problem in the synthesis of biochips. After scheduling and resource binding, the detail locations of modules should be finalized, and the locations of pins should be generated as the input to the droplet routing problem. It directly affects the hardness of the routing problem, and thus very important to a correct execution of the bio-assay. Most of previous works are droplet-routing-oblivious, which may greatly affect the routability. Meanwhile, no method has been proposed for the cross-referencing biochips. In this chapter, our method of solving the placement problem in cross-referencing biochips will be discussed. Our method is droplet-routing-aware, and the properties of cross-referencing biochips are considered.

The rest of this chapter is organized as follows. A formal formulation of the placement problem will be given in Section 4.2. An overview of our method is introduced in Section 4.3. Then, the detail of each component of our method will be discussed in Section 4.4, 4.5 and 4.6. The experimental results will be given in

Section 4.7. Finally, Section 4.8 concludes work in this chapter.

4.2 Problem Formulation

In the architectural synthesis of DMFB, module placement is the step after task scheduling and resource binding. It can be viewed as a 3D-packing problem where the third dimension is time. The starting time of each operation corresponds to the z-value of the bottom plane of a specific module. Since the module size and time span has been determined during task scheduling and resource binding, the problem can be reduced to a series of 2D-packing problems at different time intervals. After the placement, all the droplets on chip must be routed to their destination subject to some constraints. A formal description of the placement problem is given as follows:

- Input:
  - Scheduling and resource binding result
  - Chip specification, including timing constraint, chip size, optical detector number, reservoir/dispenser number
- Output:
  - Placement result, including locations of modules, reservoir and dispenser, pin locations for each route
- Objective
  - Minimize the sum of the *extended covered area*, which will be explained in the next section

4.3 Overview of the method

Our proposed placement method consists of three stages: dispenser and reservoir location generation, solving the placement problem using ILP and pin assignment.

4.4 Dispenser and Reservoir Location Generation

This is a step to find the locations of the dispensers and reservoirs. The dispenser and reservoirs should be located on the boundary of the biochip. They are not fixed before placement. However, once decided, their location is fixed during the bio-assay. Here, we do not put them into our Integer Linear Programming (ILP) formulation, but use a simple heuristic to place them on some computed positions. It is reasonable to distribute them using the proposed heuristic rather than giving them flexibility to locate arbitrarily on the boundary of the biochip, since some of them may cluster together and make the routing regions congested. More importantly, it can significantly reduce the solution space and the overheads by solving the ILP. The disadvantage is that the optimality of the solution may be hindered. But according to our experiment, the final result is mainly decided by the placement result. The algorithm is described as follows.

First of all, the waste reservoir is placed at the center of one side. A set of cells on the boundary of the biochip is then selected evenly according to the number of the reservoirs. Let there be N dispensers, we construct a graph with N vertices, where each vertex corresponds to a dispenser. For each operation in the bioassay, we check whether it is formed by two dispensers. If it is true, the weight of the edge between the two corresponding vertices will be increment by one. If no edge between them, then an edge is inserted with weight equals to one. We define the weight of a vertex as the sum of the weight of all of its incident edges. After the graph is constructed, for each of the connected component, a vertex that has the largest weight is selected and inserted into a list. Its adjacent vertices are selected and inserted to head or tail of the list in turn. Finally, the elements in the list are arranged to the boundary clockwise (or counter-clockwise), starting from the waste reservoir position. This is based on the intuition that the larger weight a vertex has, the more central it should be placed in this component, such that the distance between it and the other dispensers are evenly distributed. An example is given in Fig. 4.1. It is from the *in-vitro* benchmark in Section 4.7.

4.5 Solving Placement Problem Using ILP

We formulate an ILP to solve the module placement problem while considering the properties of cross-referencing biochip. The module placement problem in

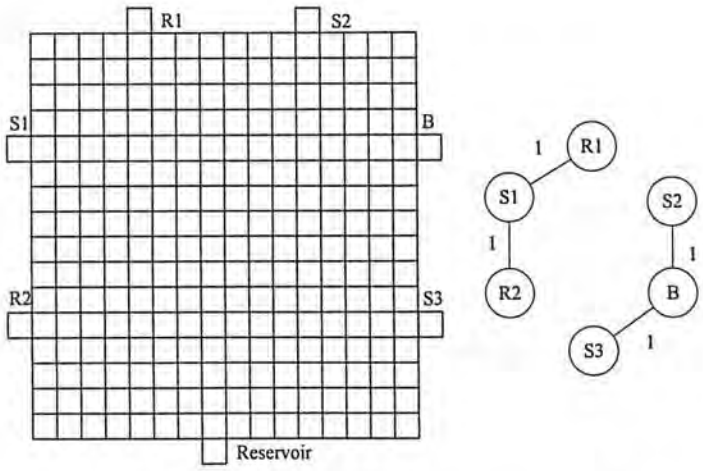


Figure 4.1: Dispenser/reservoir distribution in *in-vitro*.

Table 4.1: Notations used in ILP formulation

M^i	Module i
M^i_x, M^i_y	Lower left coordinate of module M^i
$X(M^i)/Y(M^i)$	Width/Height of a mixing module i
W/H	Width/Height of the array.
$Center(M^i)$	Center point's coordinate of module M^i
L	A large constant
A_{ij}	Extended covered area bound by module/dispenser i and j

electronic design is known to be NP-hard [24]. In order to solve it efficiently, we try to model the problem with a scalable size of variables and constraints. The core idea in the formulation is the definition of the objective function. In the ILP, the formulation is the sum of a series of *extended covered area* formed by the routes in each subproblem. The *extended covered area* of a rectangle is defined as the vertical and horizontal area span, as illustrated in Fig. 4.2. Since in cross-referencing DMFB, the droplet movement is controlled by applying different voltages to row and column. If the extended covered area of the bounding box of a route is minimized, the route is shorter. Furthermore, it helps to reduce the interference between routes. The notation used in our ILP formulation is shown in Table 4.1. Note that the index is starting from 0.

4.5.1 Constraints

4.5.1.1 Validity of modules

The modules should be inside the biochip. A further requirement is that the whole module including guarding ring should be at least one space away from the boundary of chip. This constraint is needed in order not to block the route from the dispenser nor to the reservoir. Note that this guarding ring can be shared between different modules. For module i, the above requirement is represented as:

$$M_x^i \geqslant 2 \tag{4.1}$$
$$M_x^i + X(M^i) \leqslant W - 2 \tag{4.2}$$
$$M_y^i \geqslant 2 \tag{4.3}$$
$$M_y^i + Y(M^i) \leqslant H - 2 \tag{4.4}$$

4.5.1.2 Non-overlapping and separation of Modules

For modules that co-exist at some time, they must not overlap with each other. Furthermore, there should be a separation cell around each module. For a pair of module M^i and M^j which co-exist at the same time, we have the following constraints:

$$(M_x^j + X(M^j) < M_x^i - 1) \vee \tag{4.5}$$
$$(M_x^i + X(M^i) < M_x^j - 1) \vee \tag{4.6}$$
$$(M_y^j + Y(M^j) < M_y^i - 1) \vee \tag{4.7}$$
$$(M_y^i + Y(M^i) < M_y^j - 1) \vee \tag{4.8}$$

Note that the *or* constraint cannot be directly supported in a linear program. Hence we need to *linearize* them. For this particular constraint, two more binary variables are introduced. Let them be c_1 and c_2, the previous constraint is transformed into:

$$M_x^i - M_x^j - X(M^j) + L(c_1 + c_2) > 1 \tag{4.9}$$
$$M_x^j - M_x^i - X(M^i) + L(c_1 + 1 - c_2) > 1 \tag{4.10}$$
$$M_y^i - M_y^j - Y(M^j) + L(1 - c_1 + c_2) > 1 \tag{4.11}$$
$$M_y^j - M_y^i - Y(M^i) + L(2 - c_1 - c_2) > 1 \tag{4.12}$$

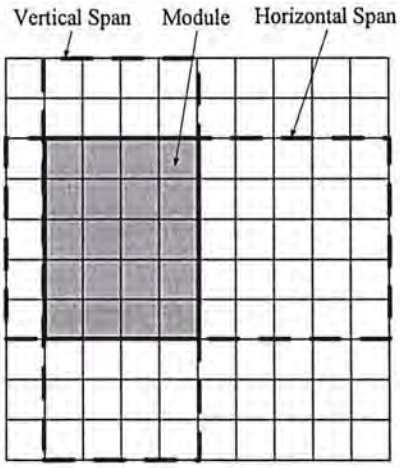


Figure 4.2: Illustration of extended covered area.

where L is a large enough number (e.g., $W \times H$). It can be seen that among the four inequalities, only one of them will be left when the value of c_1 and c_2 are decided, others will be automatically satisfied due to the large value of L.

4.5.1.3 Droplet-Routing length constraint

The maximum length of droplet paths needs to be controlled, because large value of the routing path lead to a long routing time, which may cause timing constraint violation. Furthermore, long routing length is prone to be blocked by other blocks on the way. Hence, it is necessary to have a constraint to limit the routing length to be smaller than the timing constraint. Nevertheless, since we are not computing the actual routing path during the placement phase, we will allow some degree of excess, i.e., relaxation of maximum routing length. For a routing path to be completed between two modules M^i and M^j, the relaxation of maximum routing length is determined by the module sizes. Without loss of generality, suppose M^i locates in lower left relative to M^j, source pin and sink pin are generated around M^i and M^j respectively. In the worst case, the droplets have to departure from the lower left corner of M^i, and arrives at the upper right corner of M^j. Hence, the relaxation value is set to $X(M^j) + Y(M^j) + d$, where d is a fixed parameter that can be tuned to allow extra relaxation for a congested layout area (recall that the droplet may encounter blockages during routing).

Then, the constraint is modeled as follows:

$$M_x^j - M_x^i + M_y^j - M_y^i + X(M^j) + Y(M^j) + d <= T \tag{4.13}$$

where T is the timing constraint of the biochip. This constraint help us to retain routability in the final placement result.

4.5.1.4 Optical detector resource constraint

For a set D of modules that are bound to the same optical detector, we have:

$$(M_x^i, M_y^i) = (M_y^f, M_x^f), i \in D \tag{4.14}$$

where M^f is a module in D that first appear on the time line.

4.5.2 Objective

We use the bounding box as an estimation of the routes generated. Suppose that there is an operation, in which droplet i introduced by some module M^i is needed to be routed somewhere to form as input to M^j, we use the center points of both modules to form a bound box to model the route. Let $Center(M^i) = (x_i, y_i)$ and $Center(M^j) = (x_j, y_j)$, two pairs of variables, (x_{ll}, y_{ll}) and (x_{ur}, y_{ur}), are introduced to denote the bottom left corner and upper right corner of the bounding box formed by both center points.

$$\begin{array}{rcl rcl} x_{ll} & \leqslant & x_i, & x_{ll} & \leqslant & x_j \\ y_{ll} & \leqslant & y_i, & y_{ll} & \leqslant & y_j \\ x_{ur} & \geqslant & x_i, & x_{ur} & \geqslant & x_j \\ y_{ur} & \geqslant & y_i, & y_{ur} & \geqslant & y_j \end{array} \tag{4.15}$$

Then, we can compute the extended covered area bound by this route, denoted as A_{ij}:

$$A_{ij} = W(x_{ur} - x_{ll}) + H(y_{ur} - y_{ll}) \tag{4.16}$$

The objective function is the sum of all the extended covered area:

$$min : \sum_k A_k \tag{4.17}$$

where $A_k, k = 1, 2, ...$ is a set of all the extended covered area in the subproblems.

4.5.3 Problem Partition

Note that for some bioassays, there might be a large number of subproblems. When modeled as one ILP, the number of variables and constraints may be so huge that it is impossible to solve it efficiently. For example, there can be up to two thousands of variables and five thousands of constraints in the largest benchmarks that we use. Hence, it is necessary to control the problem size. The whole problem can be partitioned into manageable problem sub-sets if the original problem set is too large. An ILP can be set up and solved for problem sub-set, and the ILP result from the previous problem sub-sets should be input as constraints for the current problem sub-set. Different partitioning strategies can be adopted here. In our implementation, we adopt a simple partitioning scheme that regularly divides the original problem set into sub-sets of the same size. Each sub-set may contain seven to fifteen subproblems according to the biochip size and the number of total subproblems in the benchmark. This strategy is simple yet working well according to the experimental result.

4.6 Pin Assignment

After the exact locations of the modules are obtained by solving the above ILP formulation, the result is not complete enough for routing. According to the placement result from [7], we summarize the following rules:

- Mixing module is modeled as a 3-pin net;
- Dilute module is modeled as two 2-pin nets;
- Optical detection and storage are modeled as 2-pin nets.

The source pin and sink pin should locate around the bounding cells of the guarding ring of a module. Then, we have a set of possible locations for any specific pin. We will use a maze routing algorithm to help us to determine the best pin location that have a shorter routing distance. In particular, for a source pin and a sink pin, we have a set of possible source pin locations and a set of possible sink pin locations. We perform maze routing for each pair of the source and pin location, and select the pair that has the shortest routing distance as the final pins. Note that the pin locations should satisfy the fluidic constraint. For example, two

Table 4.2: Running CrossRouter upon placement [7] and upon our placement

Name	# sub.*	Size	Routing result of [7]			Routing result of our placement			Improvement			
			Max/Avg cycle	Stalling Steps	# Cell Used	Max/Avg cycle	Stalling Steps	# Cell Used	Max cycle	Avg cycle	Stalling Steps	# Cell Used
in-vitro_1	11	16 × 16	20/12.09	26	246	17/9.55	9	148	15%	21%	65%	40%
In-vitro_2	15	14 × 14	20/11.07	37	254	15/5.13	3	84	25%	54%	92%	67%
Protein_1	64	21 × 21	20/15.63	49	1668	20/11.66	33	925	0	25%	33%	45%
Protein_2	78	13 × 13	20/9.86	42	976	19/8.14	31	662	5%	17%	26%	32%
								Average	11%	29%	54%	46%

*Number of subproblems.

source pins are not allowed to be adjacent. Otherwise, error may occur if two droplets are coming out from the two pins in the same subproblem.

4.7 Experimental Results

To evaluate the our placement method, comparison will be made between our work and the method proposed in paper [7]. Specifically, we will use the proposed CrossRouter to route on the placement generated by [7] and on the placements generated by our approach. Then, the solution quality is compared. Our placement method is implemented in C++ programming language, and the ILP part is solved using lp_solve 5.5 [40] and run on a 2.4GHz Intel Core II Duo machine with 1.5GB memory. We limit the ILP solver to run in a given amount of time. The time limit is set to 30 minutes here, since from our empirical experiments, the quality of the solution is almost fixed and almost no increase.

Table 4.2 summarizes the comparisons of our placement method and paper [7]. In the experiment, we define the Stalling Step (SS.) as the total number of stalling in the route that a droplet has taken. Stalling means that at a certain step, a droplet has to wait for the path to be cleared, either for waiting some other droplet to pass through, or by queueing outside the waste reservoir, etc. The root reason of stalling is from the fluidic constraint, or more possibly, from the electrode constraint for cross-referencing routing. For comparison, we run the router on the placement generated by [7] and on the placements generated by our approach. We can see from Table 4.2 that by using the placement result generated by our proposed method, the Max/avg cycles have been improved and the Stalling steps are reduced. Moreover, it is shown that the number of cell used has been reduced by more than a half comparing with the routing result of [7]. According to the

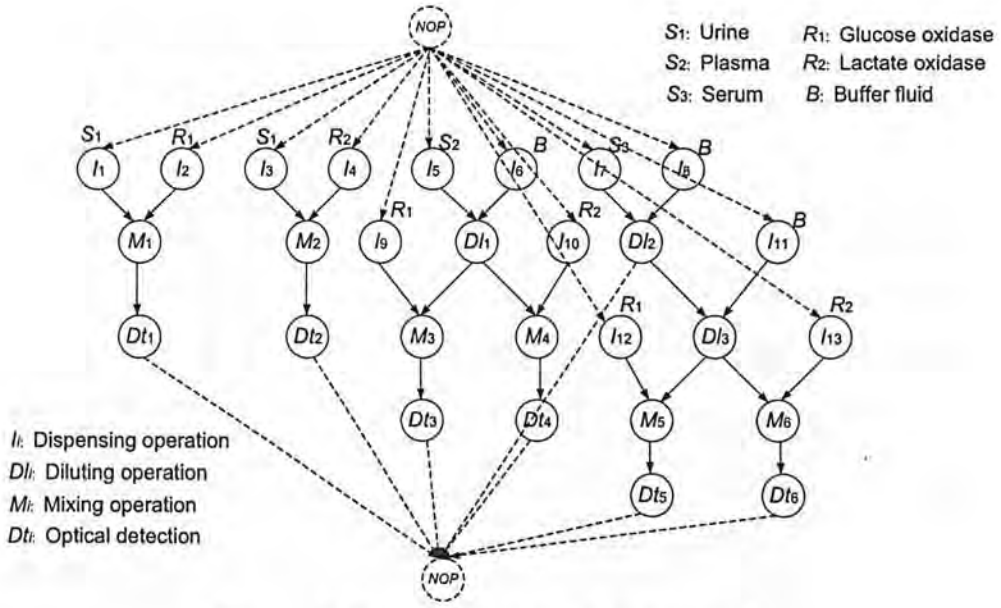


Figure 4.3: Sequencing graph of *in-vitro* [5].

experiment result of Max/avg cycle, we can see that the router can finish routing in a relative shorter time, which indicates shorter route and less congestion are achieved. The experimental result demonstrates that our placement result can give better routability and simpler routing configurations to the router.

Here we use one instance of the *in-vitro* diagnostics as an example to illustrate the proposed method. Figure 4.3 and Figure 4.4 are the sequence graph and the scheduling result of the multiplexed *in-vitro* diagnostics with 3 samples, 2 reagents and 1 buffer. As in [7], we also assume that for each type of sample, reagent and buffer, there is one dedicated on-chip reservoir/dispenser. Size of the modules are determined in the previous design stage.

Due to the extended covered area constraint, the ILP formulation gives a highly compacted placement and shorter route between droplets. Firstly, our approach saves more space on chip, which might be wasted if modules are loosely placed. Secondly, our approach is beneficial to cross-referencing routing, since the shorter a route is, the less possible that a droplet might impose electrode constraint on others (remember that the activation of electrodes are in a row-column manner).

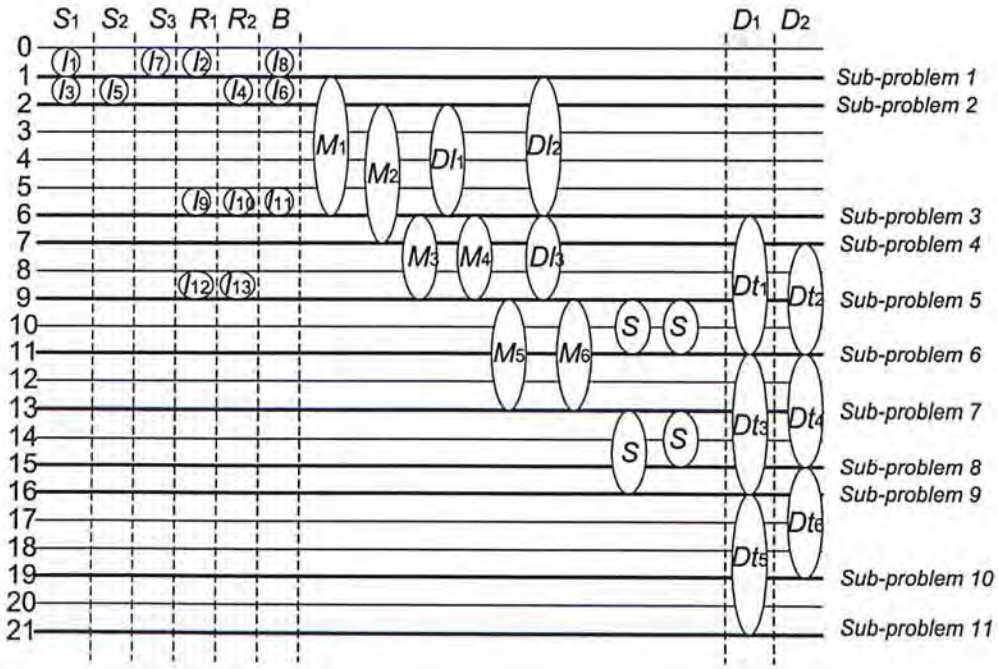


Figure 4.4: Schedule result obtained from paper [5].

4.8 Conclusion

In this chapter, we have presented an ILP-based method to solve the placement problem in cross-referencing digital microfluidic biochip. To explore the properties of cross-referencing and utilize them to optimize the placement result is the major motivation of our work. To achieve this, a three step method is proposed. At first, the locations of reservoirs and dispensers are decided using a heuristic-based algorithm. Secondly, the placement problem that we are solving is modeled as an ILP, and is solved by an ILP solver. At last, the pin location will be assigned around the modules. In a comparison with the latest previous work by running our router on the placement result generated by our method and those generated by the latest work, average improvements of 11%/29%, 54% and 46% in the max/average routing time, number of stalling steps and cell usage can be achieved. The experimental results suggest that the proposed method generates placement result that is more suitable for cross-referencing biochip routing than that by the previous work.

□ End of chapter.

Chapter 5

Conclusion

This thesis has presented design automation techniques for the droplet routing problem and the placement problem in biochips. In contrast to most of the previous works, we focus on a kind of biochips called cross-referencing biochip, in which the electrodes are addressed in a row-column manner. Electrode interference is the most important issue in this type of biochips. The interference introduces stringent electrode constraint that hampers parallel movements of multiple droplets if not designed carefully. Thus, design automation techniques are strongly needed to satisfy the constraint and to avoid incorrect movements of the droplets. By studying the characteristic of cross-referencing biochips, we propose a method for the droplet routing problem and a solution to the placement problem in the synthesis of cross-referencing digital microfluidics biochips.

For the droplet routing problem, we proposed a systematic routing method called CrossRouter to solve the problem. The nets in the problem are first sorted according to their mutual relationships. Each net is then routed using a weighted maze routing. Fluidic constraint and electrode constraint are handled during the routing step. Particularly, the stringent electrode constraint can be detected and addressed using an elegant two-coloring graph approach. Our method can route all the subproblems in each benchmark within the timing limit, while the previous work failed at some of them. The number of cells used is also minimized in our routing solution for fault-tolerance purpose. The experimental results suggest that the proposed router can solve the problem efficiently and generate good quality solutions. Compared with previous works, our method can achieve an average

improvement of 4% in routing time and 58% in runtime.

To further improve the routing result, we proposed an ILP-based method to solve the placement problem. The objective is to generate better routing inputs for the router. In this thesis, we have presented a placement method that utilizes the property of cross-referencing DMFB and generates placement results that are more suitable for the router. The location of reservoirs are first generated using a heuristic-based algorithm. Then, the placement problem is formulated as an ILP and is solved using an ILP solver. The properties of cross-referencing is considered and modeled in the objective function of the ILP. Finally, the pin locations will be generated and served as input to the later routing problem. We performed the experiments by running our router on the placement results generated by our method and those generated by the latest work. Results show that average improvements of 11%/29%, 54% and 46% in the max/average routing time, number of stalling steps and cell usage can be achieved, which demonstrates the good quality of our placement solutions.

In conclusion, by combining the effort of the placement and router tools, the flexibility of cross-referencing biochip can be further exploited. The design and implementation of such chips can also be simplified.

□ End of chapter.

Bibliography

[1] T. Xu and K. Chakrabarty, "Droplet-trace-based array partitioning and a pin assignment algorithm for the automated design of digital microfluidic biochips," in *Proceedings of the 4th International Conference Hardware/Software Codesign and System Synthesis*, 2006, pp. 112–117.

[2] T. Xu and K. Chakrabarty, "Broadcast electrode-addressing for pin-constrained multi-functional digital microfluidic biochips," in *Proceedings of the 45th Annual Design Automation Conference*, 2008, pp. 173–178.

[3] E. Griffith, S. Akella, and M. Goldberg, "Performance characterization of a reconfigurable planar-array digital microfluidic system," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 25, no. 2, pp. 345–357, 2006.

[4] T. Xu and K. Chakrabarty, "A cross-referencing-based droplet manipulation method for high-throughput and pin-constrained digital microfluidic arrays," in *Design, Automation & Test in Europe Conference & Exhibition*, 2007, pp. 1–6.

[5] F. Su, W. Hwang, and K. Chakrabarty, "Droplet routing in the synthesis of digital microfluidic biochips," in *Proceedings of the Conference on Design, Automation and Test in Europe*, vol. 1, 2006, pp. 1–6.

[6] M. Cho and D. Pan, "A high-performance droplet router for digital microfluidic biochips," in *Proceedings of the 2008 International Symposium on Physical Design*, 2008, pp. 200–206.

[7] P. Yuh, C. Yang, and Y. Chang, "Placement of defect-tolerant digital microfluidic biochips using the T-tree formulation," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 3, no. 3, Article 13, 2007.

[8] M. Pollack, A. Shenderov, and R. Fair, "Electrowetting-based actuation of droplets for integrated microfluidics," *Lab on a Chip*, vol. 2, no. 2, pp. 96–101, 2002.

[9] E. Verpoorte and N. De Rooij, "Microfluidics meets MEMS," *Proceedings of the IEEE*, vol. 91, no. 6, pp. 930–953, 2003.

[10] T. Schulte, R. Bardell, and B. Weigl, "Microfluidic technologies in clinical diagnostics," *Clinica Chimica Acta*, vol. 321, no. 1-2, pp. 1–10, 2002.

[11] V. Srinivasan, V. Pamula, M. Pollack, and R. Fair, "Clinical diagnostics on human whole blood, plasma, serum, urine, saliva, sweat, and tears on a digital microfluidic platform," in *Proc. μTAS*, 2003, pp. 1287–1290.

[12] A. Guiseppi-Elie, S. Brahim, G. Slaughter, and K. Ward, "Design of a subcutaneous implantable biochip for monitoring of glucose and lactate," *IEEE Sensors Journal*, vol. 5, no. 3, pp. 345–355, 2005.

[13] R. Fair, A. Khlystov, T. Tailor, V. Ivanov, R. Evans, P. Griffin, V. Srinivasan, V. Pamula, M. Pollack, and J. Zhou, "Chemical and biological applications of digital-microfluidic devices," *IEEE Design & Test of Computers*, vol. 24, no. 1, pp. 10–24, 2007.

[14] E. Ottesen, J. Hong, S. Quake, and J. Leadbetter, "Microfluidic digital PCR enables multigene analysis of individual environmental bacteria," *Science*, vol. 314, no. 5804, pp. 1464–1467, 2006.

[15] Y. Zhao and S. Cho, "Microparticle sampling by electrowetting-actuated droplet sweeping," *Lab on a Chip*, vol. 6, no. 1, pp. 137–144, 2006.

[16] V. Srinivasan, V. Pamula, and R. Fair, "An integrated digital microfluidic lab-on-a-chip for clinical diagnostics on human physiological fluids," *Lab on a Chip*, vol. 4, no. 4, pp. 310–315, 2004.

[17] K. Chakrabarty, "Design Automation and Test Solutions for Digital Microfluidic Biochips," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 1, pp. 4–17, 2010.

[18] Global In Vitro Diagnosis Market Analysis PRLog Free Pree Release. http://www.prlog.org/10080477-global-in-vitro-diagnostic-market-analysis.html.

[19] T. Xu, "Optimization tools for design of reconfigurable digital microfluidic biochips," Ph.D. dissertation, Duke University, 2008.

[20] P. Yuh, S. Sapatnekar, C. Yang, and Y. Chang, "A progressive-ILP based routing algorithm for cross-referencing biochips," in *Proceedings of the 45th annual Design Automation Conference*, 2008, pp. 284–289.

[21] F. Su and K. Chakrabarty, "Architectural-level synthesis of digital microfluidics-based biochips," in *IEEE/ACM International Conference on Computer Aided Design*, 2004, pp. 223–228.

[22] M. Sarrafzadeh and C. Wong, *An introduction to VLSI physical design*. McGraw-Hill Higher Education, 1996.

[23] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 1, pp. 477–521, 1987.

[24] M. Garey and D. Johnson, *Computers and intractability: A Guide to the Theory of Computers and Intractability*. WH Freeman and Company, San Francisco, Calif, 1979.

[25] F. Su and K. Chakrabarty, "Design of fault-tolerant and dynamically-reconfigurable microfluidic biochips," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2005, pp. 1202–1207.

[26] F. Su and K. Chakrabarty, "Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips," in *Proceedings of the 42nd Annual Conference on Design Automation*, 2005, pp. 825–830.

[27] F. Su and K. Chakrabarty, "Module placement for fault-tolerant microfluidics-based biochips," *ACM Transactions on Design Automation of Electronic Systems*, vol. 11, no. 3, pp. 682–710, 2006.

[28] T. Xu and K. Chakrabarty, "Integrated droplet routing in the synthesis of microfluidic biochips," in *Proceedings of the 44th Annual Conference on Design Automation*, 2007, pp. 948–953.

[29] P. Yuh, C. Yang, and Y. Chang, "Placement of digital microfluidic biochips using the T-tree formulation," in *Proceedings of the 43rd annual Design Automation Conference*, 2006, pp. 931–934.

[30] K. Bazargan, R. Kastner, and M. Sarrafzadeh, "Fast template placement for reconfigurable computing systems," *IEEE Design & Test of Computers*, vol. 17, no. 1, pp. 68–83, 2000.

[31] P. Yuh, C. Yang, Y. Chang, and H. Chen, "Temporal floorplanning using 3D-subTCG," in *Proceedings of the 2004 Asia and South Pacific Design Automation Conference*, 2004, pp. 725–730.

[32] P. Yuh, C. Yang, and Y. Chang, "Temporal floorplanning using the T-tree formulation," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, 2004, pp. 300–305.

[33] K. Bohringer, "Optimal strategies for moving droplets in digital microfluidic systems," in *Proceedings of 7th International Conference on Micro Total Analysis Systems*, 2003, pp. 591–594.

[34] K. Bohringer, "Towards optimal strategies for moving droplets in digital microfluidic systems," *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1468–1474, 2004.

[35] K. Bohringer, "Modeling and controlling parallel tasks in droplet-based microfluidic systems," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 25, no. 2, pp. 334 – 344, 2006.

[36] P. Yuh, C. Yang, and Y. Chang, "BioRoute: A network-flow based routing algorithm for digital microfluidic biochips," in *Proceedings of the 2007 IEEE/ACM International Conference on Computer-aided Design*, 2007, pp. 752–757.

[37] E. Griffith and S. Akella, "Coordinating multiple droplets in planar array digital microfluidic systems," *The International Journal of Robotics Research*, vol. 24, no. 11, pp. 933–949, 2005.

[38] Z. Xiao and E. F. Young, "Crossrouter: A droplet router for cross-referencing digital microfluidic biochips," in *Proceeding of the 15th Asia and South Pacific Digital Automation Conference*, 2010, pp. 269 – 274.

[39] Z. Xiao and E. F. Young, "Droplet-routing-aware module placement for cross-referencing biochips," in *Proceeding of the 19th International Symposium on Physical Design*, 2010, pp. 193–199.

[40] http://lpsolve.sourceforge.net.